T—Y—N—E—&—W—E—A—R

TYNE & WEAR

ATARI 8-BIT USER GROUP

Newsletter of TWAUG

Software
Editorial
Buy & Sell
Hardware

TYNE&WEAR ATARI USER GROUP

Reviews
Help line
Section
Repair Info

Public Domain Library

ISSUE #15

MAY/JUNE 1995

ATARI 8-BIT USER GROUP

U—S—E—R—G—R—O—U—P

BRING YOUR EIGHT UP TO DATE with power products from COMPUTER SOFTWARE SERVICES

THE BLACK BOX

The BLACK BOX is an add-on board for the Atari 600XL,800XL and 130XE 8-bit computers. It is a T-shaped board that plugs into the PBI port of the XL computer, or the ECI and cartridge ports of the 130XE. Connectors for both types of computers are built into the BLACK BOX so no adapter boards are necessary. A cartridge port is available on the board itself for 130XE users.

The BLACK BOX provides many unique and useful functions. The four primary functions are:-

* RS-232 serial modem port
* Parallel printer port
* SASI/SCSI hard disk port
* Operating System enhancements

The BLACK BOX is $199.95 for the basic unit, and $249.95 with an onboard 64K printer buffer.
Shipping and Handling extra.

THE BLACK BOX ENHANCER

A must for all BLACK BOX owners. The BLACK BOX ENHANCER is a plug-in module for your BLACK BOX, enhancing the printer functions and adding an instantly available, full featured sector editor!

Installation of the BLACK BOX ENHANCER requires one simple solder connection. Only $49.95 plus shipping/handling.

THE FLOPPY BOARD

Our latest and greatest product. The FLOPPY BOARD is an add-on expansion board for the BLACK BOX interface. It allows the use of the same inexpensive floppy drive mechanisms used in IBM computers. The FLOPPY BOARD is the first floppy drive interface to support "high density" floppy drive mechanisms in either 5.25 inch or 3.5 inch. Built into the FLOPPY BOARD are our BLACK BOX ENHANCER and a version of our SUPER ARCHIVER to allow copying of protected disks for 3.5 inch format. Included with the FLOPPY BOARD is our program to read and write to IBM or ST formatted disks. This makes the FLOPPY BOARD the best way to transfer files to and from your 8-bit.

The FLOPPY BOARD is only $149.95 plus shipping & handling.

THE MULTIPLEXER

This device brings the power and flexibility of larger systems to your 8-bit. The Multiplexer is a collection of cartridge interface boards that allow up to 8 Ataris to read and write to the same drives (typically a hard disk), access the same printer(s), and talk to each other. It is the first practical networking system for the Atari 8-bit computer.
One "master" computer (any 8-bit) is equipped with the master Multiplexer interface.

Then up to 8 "slave" computers hook up to this master, each having their own slave interface. The "common" peripherals (things that are to be shared) are connected to the master. On each slave, all disk and printer I/O is routed through the master, so no extra disk drives are needed.

The Multiplexer sells for $199.95 for a master and two slave units with cable. Additional slave units are $89.95 each, plus shipping/handling.

THE SUPER ARCHIVER II

The SUPER ARCHIVER II edits and copies all enhanced density programs plus retains all the features of the SUPER ARCHIVER.

The SUPER ARCHIVER II is only $99.95 plus shipping $ handling. NOTICE: if you already have THE SUPER ARCHIVER you may upgrade to S.A.II for only $29.95 plus shipping/handling. Software only.

THE BIT WRITER

The Super Archiver BIT WRITER is capable of duplicating even the "uncopyable" Electronic Arts and Synapse Syn-series, which employ 34 full sector tracks. The BIT WRITER must be used with the SUPER ARCHIVER

The BIT WRITER is only $79.95 plus shipping/handling.

THE ULTRA SPEED PLUS OS

The Operating System that should be in every XL/XE computer! The Ultra Speed Plus puts unbelievable speed and convenience at your fingertips.

Use any DOS to place Ultra Speed formats on your disks (with XF551 or modified 1050 drives), reading and writing at this speed with most programs. This high speed mode can be turned off for maximum compatibility.

Four simple solder connections are required for installation if your machine has a socketed OS ROM. The Ultra Speed OS is only $69.95 plus shipping/handling.

For more information on these and other 8-bit products:

CONTACT
COMPUTER SOFTWARE SERVICES
PO BOX 17660
ROCHESTER, NEW YORK 14617
USA

ORDERING LINE: (716) 429-5639
FAX: (716) 247-7158
BBS: (716) 247-7157

TWAUG NEWSLETTER

EDITORIAL

Who to blame!!!

John Matthewson
David Ewens
Max Gerum

Please accept our apology for any delay in sending out PD orders and also to any subscriber who received a second reminder for subscription renewal. All this upset was due to a postal dispute in the sorting offices in Newcastle. Upto now everything seems to be settled again but there is still a backlock of mail lying somewhere, hopefully the mail delivery will be back to normal soon.

The contribution fees for home and abroad:

HOME	1 COPY	£2.00
-DO-	6 COPIES	£11.00
EUROPE	1 COPY	£2.20
--DO--	6 COPIES	£12.50
ELSEWHERE	1 COPY	£2.50
---DO----	6 COPIES	£14.00

REMINDER:

The book--The Complete and Essential Map--anyone unable to pay the full amount of the cash price can pay in instalments, whenever and whatever you can afford, at no extra charge.

The next issue will be ready by mid-July.

CONTENTS

EDITORIAL	3
IMMEDIATE MODE by John Picken	4
BASIC PROGRAMMING HELP by Hawke	7
HINTS & TIPS	9
ATARITECH BBS Null Modem Adapter	11
MAKING SIGNS WITH YOUR 8-BIT by Thomas J.Andrews	12
XF551 ENHANCEMENT Review by David Bryant	17
XIO EXPLAINED by Paul Hollins	18
COPING CAPERS by Andrew C.Thompson	22
S.A.M.S A thanks to our helpers	24
LACE ADVERT	24
CRACKING THE CODE by Keith Mayhew	25
ADVENTURES REVIEW	31
ATARI BOOK LENDING LIBRARY A new idea by Nir Dary	33
FOR SALE SECTION	33
SWAPPING SECTION	34
DISK CONTENT	34
DEVICE HANDLERS by David A. Paterson	36
FAMOUS QUOTES	38
MICRO DISCOUNT	39
ACPC	40
CURRENT NOTES & PHOENIX	40

IMMEDIATE MODE

by John Picken

For several years I experimented with the power available in BASIC's Immediate Mode--you know, using BASIC to carry out operations without a program. Most users are aware that you can enter something like PRINT 5+6 and have the answer pop up on the screen without affecting any program you might be working on. Many of those same users are unaware that this technique can be extended well beyond simple operations.

It is possible to use Immediate Mode to carry out nearly all operations provided by the Disk Utility Package and even have access to "pop-up" information screens. With the extra power available in Turbo BASIC, the user can also, easily, copy files, do number base conversions, move memory, and perform bit-wise logic functions.

Immediate Mode is subject to the same line length limitation as are programs. This means a maximum of three screen lines (i.e. 120 bytes with margins set at 0 and 39). The real power of this mode occurs in the fact that the line may be ENTERed from a disk file--it need not be typed in each time you want to use it.

PROGRAMMING IN IMMEDIATE MODE

As a programming exercise, fitting the maximum number of functions into such a short file is both challenging and instructive. First obviously, you have to know the acceptable abbreviations for any functions you wish to use and you have to be aware of which spaces are mandatory. You will often have to experiment here, though if you remember some of BASIC's ground rules, this is not all that difficult.

For example a variable name can never start with a digit, but constants normally contain only digits. Therefore, no spaces are required in something like FORX=1TO99STEP11 as BASIC is smart enough to figure out where values and key words begin and end. But two spaces are definitely needed in FOR X=A TOB STEPC because BASIC needs an indication of the end of the variable names--without spaces, it thinks the variable is ATOBSTEPC.

With Turbo BASIC you will always have to experiment with abbreviations as they are not listed in the documentation. Here again, common sense makes this easier. Essentially as in Atari BASIC, you can abbreviate any key word to just after the point of ambiguity and, in a few cases like GOTO, before that point. For example, Turbo will accept REP. or RET. but you can't shorten these any further without getting REM in the output listing.

A second requirement, if you want any kind of screen display beyond simple one liners, is a knowledge of cursor and screen controls. This includes the effects of commas, tabs and semicolons in PRINT statements. For example, even in its abbreviated form, the POSITION statement takes a lot of physical space in the line, but you can often accomplish the placement with cursor controls. Here, thinking from a different approach can often save bytes--instead of POSITIONing at the bottom of the screen, arrow the cursor off the top.

LINE NUMBERS

One of the biggest difficulties the programmer has to overcome is the non-availability of all functions which require line numbers. I've read articles on the use of Immediate Mode in ANALOG, Antic, and Compute! In none of the articles was the author able to overcome the lack of TRAP or IF-THEN-GOTO. A standard one-liner, presented in all three magazines, involved reading a disk directory from BASIC. Every approach to this I've ever seen terminates in an "End of file" error condition. The code in Example A is typical of those which gave a two-column listing (two columns being the reason for the POKE and for the comma after the ?F$ statement).

IMMEDIATE MODE continued

Example A (Atari BASIC Directory)

```
CLR:DIMF$(20):POKE82,0:?"c":CL.#1:
O.#1,6,0,"D1:*.*":F.X=1TO65:I.#1,F$:?F
:N.X
```

Example B is the same code spaced and unabbreviated as it would be if you had put a line number in it and then LISTed it to the screen. All examples are shown as they would appear on the screen with margins set at 0 and 39. For the sake of printing and legibility, substitutions of control characters are used and explained--in these two, "c" represents the Clear Screen symbol.

Example B (Expansion of A)

```
CLR :DIM F$(20):POKE 82,0:? "c":
CLOSE #1:OPEN #1,6,0,"D1:*.*":FOR
X=1 TO 65:INPUT #1,F$:? F$,:NEXT X
```

The source of the error is obvious: unless the directory holds 64 files, (the TO 65 is there to allow printing of the "FREE SECTORS" line), you will always get an error 136. Ending in an error, coincidentally, means you have left channel #1 open. Is there a solution without line numbers?

First look at the ways most programmers solve the problem with line numbers. Example C is more common but Example D, more often thought of by those with machine language experience, is simpler and need not be modified for use with MyDOS. Note that in Example C, the upper limit must be increased to ensure the POP actually pops the correct thing (critical if this is made into a subroutine).

Example C (Atari BASIC Directory)

```
10 CLR :DIM F$(20):POKE 82,0:? "c":
CLOSE #1:OPEN #1,6,0,"D1:*.*"
20 FOR X=1 TO 128
30 INPUT #1,F$:? F$,
40 IF F$(5,16)="FREE SECTORS" THEN
60
50 NEXT X
60 POP :POKE 82,2:? :CLOSE #1
```

Example D (Atari BASIC Directory)

```
10 CLR :DIM F$(20):POKE 82,0:? "c":
CLOSE #1:OPEN #1,6,0,"D1:*.*"
20 TRAP 50
30 INPUT #1,F$:? F$,
40 GOTO 30
50 POKE 82,2:? :CLOSE #1
```

At first glance it seems impossible to avoid the "End of file" error without the use of line numbers. As with most programming problems, often the best solution is a new approach--instead of waiting for end of file or looking for "FREE SECTORS", consider the format of all directory lines. Every line but the last returned has exactly the same format. Each starts with one of only two possible characters: the space or the asterisk. This fact is used in Example E (expanded in F) to solve the end of file problem.

Example E (Atari BASIC Directory)

```
CLR:DIMF$(20):POKE 82,0:? "c":CL.#7
:O.#7,6,0,"D1:*.*":F.X=0 TO 1:I.#7
,F$:?F$,:F=F+1:X=F$)"/":N.X:?"Total
:";F-1:POKE82,2:END
```

Example F (Expansion of E)

```
CLR :DIM F$(20):POKE 82,0:? "c":CLO
SE #7:OPEN #7,6,0,"D1:*.*":FOR X=0
TO 1:INPUT #7,F$:?F$,:F=F+1:X=F$)"/
":NEXT X:?"Total: ";F-1:POKE 82,2:
END
```

The key to the routine is the X=F$)"/" statement. The only time this condition can be true (logic value of 1 assigned to X) is when F$ contains the "FREE SECTORS" line which starts with a digit. Until this happens, the loop counter is continually reset to 0 so that the TO condition is not met and the loop repeats.

There are other advantages in this method: It need not be modified for use with SpartaDOS as there is no true upper limit to the loop. It works fine with MYDOS as it doesn't check the string for "FREE SECTORS" which is one place further right due to the four digit sector counts. There is even sufficient room to have the routine count and print the number of directory entries. Finally, assuming no disk errors, it will restore the left margin and END to close all channels on completion.

IMMEDIATE MODE continued

CHANNEL #7

There is yet one more advantage in that the routines use IOCB #7, the one reserved for BASIC. You can use #7 as long as you close it first (it was opened and possibly left open, in mode 4, by BASIC in response to your ENTER command). So what's the advantage in this? It's practically 100% safe in case you leave it open whether by hitting Break or otherwise forgetting it. With #7, this doesn't matter--when BASIC wants to use it BASIC will first close it. There is only one time you can get an error when #7 is left open whether by you or by BASIC. This error occurs on the first LPRINT attempt though subsequent ones will work. But, you don't really use LPRINT anyway, do you (I hope)?

Example G (H) takes advantage of this fact to free some space. It also employs Turbo's more efficient commands, and drops the ? "Total: " statement though it still reports total entries. Sufficient space is gained to allow the user to specify any drive and file mask. The string dimension is also increased to allow for path names. The "i" just before :I.#16 is the Shift Insert screen control. This last has the effect of placing the cursor on the default filespec where the user may edit it before hitting Return.

```
Example G (Turbo BASIC Directory)
CLR:DIMF$(99):F$="D1:*.*":POKE82,0:
CLS:? F$;"i":I.#16,F$:CL.:O.#7,6,0,
F$:REP.:I.#7,F$:?F$,:F=F+1:U.F$)"/"
:?:POKE82,2:?F-1
```

```
Example H (Expansion of G)
CLR :DIM F$(99):F$="D1:*.*":POKE 82
,0:CLS :?F$;"i":INPUT #16,F$:CLOSE
:OPEN #7,6,0,F$:REPEAT :INPUT #7,
F$:? F$,:F=F+1:UNTIL F$)"/":? :POKE
82,2:? F-1
```

BEYOND ONE LINE

You can create longer files in Immediate Mode and the lines of the file will enter and execute sequentially like a form of batch file. You can use IF-THEN action in these lines but there are more restrictions than with single line files.

Any attempt to READ or to use INPUT (in any form, INPUT #, etc.) in a line will abort entry of the file at that line as, of course, does any error condition. To get around the restrictions, you have to become creative with Turbo's GET function, possibly using it to build an input string character by character.

Having written a few such files, I recommend writing them as normal programs first to test the algorithm. The final conversion to Immediate Mode can then be done on something that you know works. As an upper limit for complexity, I wouldn't go much beyond four or five sectors--they get incredibly difficult.

CONCLUSION

TWAUG has previously released TurboDUP which demonstrates all the techniques mentioned above. I admit that I cheated on some of the modules by sneaking in some machine language, but it's practically guaranteed not to conflict with anything as it uses only the Printer and Cassette buffers. If you want to examine the files you can load them into TextPRO to see them as is. For a better look at how they work, add line numbers and ENTER and LIST them in Turbo. With several, you'll have to split the compound lines on a colon to allow room for the line number.

If you're interested in BASIC programming, I recommend you take a look at Immediate Mode. Not only does it teach you more about BASIC, it makes you think of alternative and less conventional techniques, a habit that will serve you well in any other programming endeavour from writing batch files to assembler masterpieces.

ATARI BASIC PROGRAMMING HELPS

By Hawke,

Sysop of the Nine Hells Information Exchange of Philadelphia, (215) 785-2625

As a whole, standard Atari BASIC is not the best programming medium in existence. One of the major problems stems from the fact that it was created too many years ago. Be that as it may, new programmers are being subjected to it now, unnecessarily, and must deal with a user unfriendly language. Inspired by a set of PASCAL notes, I venture to write these helps, so that BASIC programmers can learn from my experience, as well as the Phoenix, with whom I have consulted about this text. Any questions or comments are welcomed. Contact myself or Phoenix through the above Bulletin Board number.

The following sections deal with POKE's, PEEK's, small machine language routines, some program listings, BASIC tricks, and other such nonsense.

ROUTINE ONE: (Three Lower case H's), (INVERSE asterisk), (Capital L), (Capital V), (INVERSE Lower case D)

ROUTINE TWO: (Lower case H), (INVERSE Quote), (Control P), (SPACE), (Upper case V), (INVERSE lower case D), (INVERSE control A), (INVERSE capital T), (INVERSE close parenthesis), (Control ,), (INVERSE control E), (INVERSE capital U), (Control .)

(Ed. Note: When typing in the above routines, DON'T type in the commas.)

SECTION ONE: Machine Language Routines

Many BASIC programmers are fearful of using machine language subroutines in their programs. I know this to be true, because I was one of them. I have since gotten over that fear, after collaboratively writing the P-Net 4.5 BBS system with the Phoenix, and have converted a few others.

If you have ever used Character Fonts, made with any number of font editor programs (Envision, SuperFont, etc.), you know that they can be used to create fancy text, graphics, or maps. The problem with them is that one must enter in 1,024 bytes, poke them into a page of memory, and then change the register holding the address of the ROM set to the new set. To do all this takes a bit of time to do. The following machine language subroutine loads in a character set and adjusts the register in seconds.

```
1 POKE 881,0:OPEN #3,4,128,"D:SET.
DAT": POKE 884,0:POKE 885,112:POKE
888,0: POKE 889,4:POKE 882,7
2 AA=USR(ADR("ROUTINE ONE"),48):CL
OSE #3:POKE 54277,4:POKE 756,112
```

* SET.DAT would be the font.

* Poking 885,112 tells the machine which page of memory the redefined character set will be stored in (since a character set MUST start at the beginning of a page).

* To return to the ROM set, simply POKE 756,224. To go back to the redefined font, POKE 756,112.

The next machine routine deals with speedy Disk I/O. Atari BASIC programmers know that using the INPUT and GET commands with disk files is SLOW, especially when using large amounts of data. The following machine language subroutine solves this problem by allowing you to read an entire file into a string, virtually, in seconds. This can be useful when printing out text files or editting machine language programs (by string search, etc.).

```
900 CLOSE #1:OPEN #1,4,0,"D1:filena
me.ext"
1000 CE=INT(MEM/256):POKE 857,CE:
POKE 856,MEM-256*CE:BUF$(MEM)=" "
:BOF=C1
1010 POKE 853,INT(ADR(BUF$)/256):
POKE 852,ADR(BUF$)-PEEK(853)*256
1020 DUM$="ROUTINE TWO":POKE 850,
7:IF USR(ADR(DUM$))=136 THEN BOF=
C0
1030 X=PEEK(856)+256*PEEK(857):BU
F$=BUF$(1,X):RETURN
```

ATARI BASIC PROGRAMMING HELPS cont.

* MEM is the amount of characters that BUF$ is dimensioned to hold.

* In line 1000, the routine divides up that number into two numbers with a formula (16-bit conversion). This tells the subroutine the maximum length the string can hold. It then sets the last character in BUF$ equal to a character, thus, it defines BUF$ as being full.

* Line 1010 divides the address (starting location) of BUF$ by the same 16-bit conversion, and stores it for the subroutine, so that it knows where to start putting the incoming data.

* Line 1020 executes the machine routine, and reads in the data, checking for an End Of File (EOF) error; #136. If such an error is met, the variable BOF is set to zero.

* Line 1030 retrieves the number of bytes long the routine loaded, and tells BUF$ that it is X number of characters long.

* If BOF=0, the user may account for the EOF and jump back to the routine after clearing BUF$.

* By using this routine, one can copy files through BASIC. Simply, after reading in a file, print the string to disk.

```
10 CLOSE #1:OPEN #1,8,0,"D1:destin
at.ion":PRINT #1;BUF$:CLOSE #1
```

The last machine language routine is a clock. The Atari has a built in clock and doesn't even use it. It follows the electrical current in the U.S., which flows at 60Mhz. The following routine, written by the Phoenix, sets up a clock, using page 6 memory (1536) to store the data, and operates during each VBI (Vertical Blank Interrupt). You mere-ly have to supply the current hour, minutes, seconds, and 60ths of a second. You can retrieve the current time by merely using PEEK commands.

The only stricture is that if you use a graphics command (eg. GRAPHICS 0) you must re-initialize the clock.

```
32700 FOR A=1 TO 105:READ B:POKE
1535+A,B:NEXT A
32705 ? "Enter Military Time as Hou
rs,Minutes,Seconds,60ths": INPUT H,
M,S,SS
32710 X=USR(1611,H,M,S,SS)
32720 DATA 238,115,6,173,115,6,201,
60,240,3,76,95,228,169
32730 DATA 0,141,115,6,238,114,6,17
3,114,6,201,60,240,3,76,95,228,169,
0,141
32740 DATA 114,6,238,113,6,173,113,
6,201,60,240,3,76,95,228,169,0,141,
113,6
32750 DATA 238,112,6,173,112,6,201,
24,240,3,76,95,228,169,0,141,112,6,
76,95
32751 DATA 228,104,104,104,141,112,
6,104,104,141,113,6,104,104,141,114
,6,104,104,141
32752 DATA 115,6,169,6,170,160,0,76
,92,228,0
```

* The PEEK locations for the time are:

Hours	- 1648
Minutes	- 1649
Seconds	- 1650
60ths	- 1651

* The program first pokes the machine language subroutine into page 6 memory, and then prompts you to enter the time.

* It then GOSUBS the machine subroutine, sending with it the values of the current time, and then starts the clock.

* After a GRAPHICS command, just PEEK in the values of the correct time into H, M, S, and SS, and reintialize the clock.

SECTION TWO: BASIC Tricks

Since Atari BASIC hasn't many special features, one must compensate by "tricking" the system. These "tricks" are actually ways around BASIC that speed up processing, format output, etc.

ATARI BASIC PROGRAMMING HELP cont.

The first trick deals with filling a string with one character. To fill a string with 1,000 spaces, one would usually do this:

```
10 DIM L$(1000)
20 FOR A=1 TO 1000
30 L$(A,A)=" "
40 NEXT A
```

This method, though fine programming, is SLOW. The trick around it is a quirk in the BASIC; well, not actually a "quirk", but a hidden back door. The line below is the aforementioned routines equivalent.

```
10 DIM L$(1000):L$=" ":L$(1000)=" "
:L$(2)=L$
```

Not only is this routine shorter, but it's almost instantaneous, depending on the size of the string. I find a one second wait for strings in the xx,xxx digit range, but I think that it's worth the wait.

SIDE NOTE: If you DIMension a string at 1000, and aren't filling the ENTIRE string with one character, say 500 from a 1,000 character string, and use the formula [L$=" ":L$(500)=" ":L$(2)=L$], it will fill it with 501 spaces. To clear this up, all you need do, is put "L$=L$(1,500))" after setting it. This is true, even when filling 998 of the 1,000 characters.

The next "trick" deals with the INPUT statement. Anyone who has programmed using the INPUT statement knows that it puts a "?" on the screen to signify an input request. This can be bothersome to some, especially me, when they want to make prompts look nice. That "?" just looks outta place, and sometimes is included in the returned variable.

There are two solutions. One is to write your own "Get-Line" routine, using an open keyboard buffer, getting keypresses, adjusting for backspaces and control characters, and using up memory, OR...

Use the following Input command that is totally the equivalent of your regular INPUT.

```
INPUT #16,variable
```

That's right! An input through standard Atari BASIC without the dreaded "?". It works exactly like the standard INPUT command. Sure, you could open a channel to the editor (E:) yourself, but this method requires no additional open buffers.

Ever want to intialize a disk from BASIC? No, I don't mean just format it, I mean put DOS on it too. Well, I have good news and bad news. The GOOD news is that you can, the bad news is that that's ALL you can put on it; DOS, no DUP. In fact, the writing of DOS to the disk is as fast as writing from DOS itself!

```
OPEN #1,8,0,"D1:DOS.SYS":CLOSE #1
```

Yes, that's it! All DOS's except SpartaDOS and TOPDOS will work, even the OSA+ DOS's. You have just written DOS to your disk from BASIC.

Did you ever want to Re-Boot (Warm Boot) the computer from BASIC without hitting System Reset? Try this:

```
X=USR(58487)
```

This will "Warm Boot" the computer, but unlike the RESET key, it is the equivalent of turning the machine OFF and ON again; so you might call it an artificial COLD boot. You may hold down OPTION as this boot takes place to disengage BASIC. Its Hexadecimal number is E477. You can do it from a DOS as well.

SECTION THREE: PEEK's and POKE's

Did the speed of Atari BASIC processing ever bug you? Well, there's a way to increase data processing 33%. It will increase problem solving speed, data processing, and graphic plotting.

```
POKE 559,0
```

This POKE will shut down ANTIC, thus shutting of the screen, so that processing speed is increased. To turn it back on, use POKE 559,34.

ATARI BASIC PROGRAMMING HELPS
cont.

To find out what version of Atari BASIC you have in your computer, you simply:

PRINT PEEK(43234)

A number 32 or less is revision A. The value 64 or less is revision B; built into most XL's. And a number greater than 64 is revision C; the current version built into the XE line. Though I have an old 800XL, for some reason I have revision C built in; a blessing.

Here are some keyboard POKE's to play with:

* 755,004 - Turns character set Upside-Down, and disables Inverse video.

* 755,008 - Right-Side up character set WITHOUT Inverse video.

* 755,013 - Upside-Down characters WITH Inverse video.

* 755,010 - Right-Side up character set with Inverse video (Normal mode).

Special recognition to XLent software and David Castell for creating the fine word processor used to create this text. BRAVO.

Please send me comments concerning these notes. A positive response might warrant a Part Two.

Hawke

I downloaded this article from the Pandora BBS (614) 471-9209 on 10/22/89. It looks like it came from the Nine Hells Information Exchange of Philadelphia (PA) BBS, (215) 785-2625. I thought it might be good for those still programming in Atari BASIC. I hadn't seem most of the tips before except the one on getting rid of the question mark. I'll bet others haven't either. I hope you find this article helpful. - Grant -

HINTS & TIPS.

Atari 1027 printer tips.

If you have problems finding replacement ink rollers for your 1027, buy an inexpensive stamp pad inker (roller type) and ink the roller with it. This produces much clearer, better defined letters than even a new roller.

How to use the special 1027 features with AtariWriter.

Before using a character from the international character set, you must type (CTRL O) 27 (CTRL O) 23. Next type (CTRL O), and then the decimal number of the character you wish to use (from the chart in the 1027 printer manual).

To use the 1027's underline feature, first turn off the international character set, if you've been using it, with (CTRL O) 27 (CTRL O) 24. To start underlining, type (CTRL O) 15, to end, type (CTRL O) 14. Please note that the control character is the letter O, not the number zero.

Also, when using the 1027 with AtariWriter, select printer 3 (820) when using the print-preview or print-file commands.

A tip for disk drive users.

Always start the program you are working on with:

1 REM SAVE "D:Filename.ext"

Then, as you type in the program, you can delete the line number and the REM to save your work to the disk, and not forget the filename you were using. This saves a lot of grief when that old ATARI BASIC bug creeps in to your work and locks up the keyboard. I recommend saving the program every 100 line numbers.

T.W.A.U.G.
P.O.BOX No.8
WALLSEND
TYNE & WEAR
NE28 6DQ
Phone: 0191-2710086

ATARITECH BBS! Null Modem Adapter

Have you ever tried to transfer files from one computer to another? You probably connected two modems together, or called yourself if you happen to have two telephone lines. It was probably a very slow process, unless you happen to own TWO 9600 baud modems! A Null-Modem Adapter can help you. It will allow you to transfer files at up to 9600 baud and not have to tie up your telephone line.

A Null-Modem Adapter is simply a connector between two computers that allows direct communication between two computers. An actual modem is never used, so you can use the highest baud rate that both computers can handle. What the null-modem adapter does is convince the computers that they are connected to a modem instead of another computer.

Before building your Null-Modem adapter, you need to determine which types of connectors to use. Most null modem connectors use a male and a female DB-25 (modem type) connector. If you already have modem cables for both computers, you will probably find that a null modem connector with two female connectors will be more useful to you. This way you can connect the two modem cables together with the null-modem adapter and be ready to roll!

What you need:

Soldering Iron and Solder
Approx. 12" of #24 stranded wire
Cover shell - Shack # 276-1520
Two DB-25 solder-type connectors
Female - Radio Shack 276-1548
Male - Radio Shack 276-1547
(determine which ones you need)

How to Build It:

Full Handshake Null Modem (best):

Connector:	A		B
	1	to	1
Connect	2	to	3
These	3	to	2
Pins:	4	to	5
	5	to	4
	6,8	to	20
	7	to	7
	20	to	6,8

The pins on the connector are numbered, but remember that pins on the Male connector, looking at the solder side, narrow-edge down, are numbered right-to-left, top row first. The female connector is numbered left-to-right!

If this one does not work, and you have CHECKED the WIRING, then try using the "No-Handshake" null-modem adapter:

Connect the following pins OF EACH connector together:

Connect pins 4 + 5 together.
Connect 6, 8, + 20 together.

Connect these pins BETWEEN the two connectors together:

Connector:	A		B
	1	to	1
	2	to	3
	3	to	2
	7	to	7

How to Use the Adapter:

Boot up each computer with a good terminal program. For the Atari 8-bit I suggest AMODEM 7.5 because it can handle BOTH 9600 baud and YMODEM transfer protocol. This will give you the fastest possible data transfer. Set both terminals to the fastest baud rate that both computers can handle.
Next connect the computers together with the Null-Modem Adapter. Following the instructions of each terminal program, simply set the sending computer for upload and the receiving computer for download. Remember to use the same protocol on each computer, and it is usually better to start the receiving computer first. Basically, that's it! Easy!
If you have any questions about this or any technical questions about Atari 8-bit computers, you can call the AtariTech BBS at (813) 539-8141 We have many files on easy-to-build hardware projects, memory upgrades, fixes and mods.

MAKING SIGNS WITH YOUR 8-BIT

By Thomas J. Andrews.

This article first appeared in Current Notes vol. 13 no. 10.

My family operates an on-site retail vegetable outlet (that's a farm stand, for the politically incorrect.) on our farm here in Central New York. as you can well imagine, good signs are an important part for this, or for that matter, of any retail operation. Signs can be used to inform customers of a wide variety of things such as where to park, product identification, availability, unit of sale, price and others.

For the past few years, I've been generating many of our signs with my XL system. I've looked at several programs that can be pressed in to this service, but have settled on three for most of the work. Print Shop, Print Power and Signmaker.

My sign making hardware consists of a 256K Rambo-enhanced 800XL, XF551 and 1050 drives, 850 interface, a Corilla monochrome monitor, and two 9-pin printers, a Star Gemini 10X and an Epson LX-800. All hardware and software, except for the Rambo upgrade, XF551, and Singmaker were purchased used at garage sales and other sources at, I might add, excellent prices.

SO WHAT KIND OF SIGNS DO YOU MAKE?

We use a variety of signs for a variety of purposes. The larger, more permanent signs, like the roadside "Fresh Vegetables Just Ahead" signs, are not generated by the computer. Our stand is exposed to the weather, and we find that computer signs don't hold up well enough to be exposed to the wind and rain for long. (I do, however, have secret plans to try to generate patterns for those permanent sings when next they need replacement.)

At the stand itself, we'll frequently use banners to identify our leaders to potential customers as they look us over from the road. In retail terms, a loader is an item used to attract customers, often sold at a relatively low price. If that price is unprofitable, it becomes a lost leader.

The theory is that once customers stop for the leader, they'll buy other things, too, and the total purchase results in higher profits. You can't make any profits at all if people don't stop. We only deal in seasonal produce, so our leaders change as the season progresses.

Once the customer is at the stand, a lot of information must be made available. By tradition, different items are priced according to different criteria. Some are by count, others by weight, still others by volume, and some by appearance, and the customer needs to be informed as to which is which. Some unusual, exotic vegetables or newly developed vegetables need to be identified, and the customer needs to know what makes them better, or at least different, from what they're used to. Sometimes we'll even provide basic cooking instructions for these exotics. Other signs may proclaim a special deal for quantity purchases, or a special service we provide on order. Still others will make a special request of the customer, like "Please don't pick up the pumpkins by the stems."

That discription makes it sound like we wall-to-wall signs, and, of course, it isn't like that at all. Not all types of signs are used on all produce items simply because they aren't needed, and too many signs just get in the way.

In general, our signs need to be noticable, yet unobstrusive. They need to enhance rather than conflict with the particular display involved. As such, we need a variety of sign sizes, so that the size may be matched with the purpose. This is the biggest single variable we require.

The most important purpose if a sign is to get the message across, and this needs to be remembered at all times when designing signs. We like to have the style consistant

MAKING SIGNS WITH YOUR 8-BIT cont.

across all signs. Graphics are used only when they enhance the message, or to make the sign more noticeable. The same thing goes for borders and special effects. We use as few fonts as possible, and these are all similar to each other. Oh, there are exceptions, of course, like oriental fonts and graphics for oriental vegetables. We try very hard not to go overboard with this stuff, because the product is what is supposed to be admired, not the sign above it.

WHEW! THERE'S MORE TO THIS THAN I THOUGHT!

There sure is! It's taken us over 30 years of trial and error to learn what works best for us. That's why we need three sign making programs. No single one is versatile enough, at least not of those I've seen yet. Each one has it's strengths and weaknesses.

MAKING SIGNS WITH PRINT SHOP

Print Shop, by Broderbund Software, is probably the best known of these programs. It has four forms of output that can be used as signs: Signs, Banners, Greeting Cards, and "Screen Magic".

One big advantage of Print Shop is its graphics editor. The graphics editor on the original Print Shop can be used to change existing graphics or create new ones using keyboard, joystick, or Koala Touchpads. The Print Shop Companion, a seperate program, has an improved graphics editor along with border and font editor. Due to these editors, literally hundreds of Print Shop graphics, popularly called icons, are available from various public domain sources, (check out the TWAUG PD library for their large collection of icons, borders and fonts). Because of the existance and widespread availability of these icons, several programs have been written that use them. Antic magazine once published one that would print mailing labels with them, and one that would print more varied banners using Print Shop fonts. Unfortunately, few of these icons have a vegetable theme.

Another of Print Shop's big advantages is that it is one of the easiest 8-bit programs to learn and use. The manual is almost superfluous, and novice users can produce quality output the first time they boot up. Well-designed menus guide you through your creation, and at each step you can go back to an earlier one merely by pressing the ESCAPE key. There are few commands to learn. Ironically, that advantage is also the cause of Print Shop's biggest disadvantage. In order to make the program so easy and foolproof to operate, the designers chose to limit user options as to the forms of output. Only one font and one icon may be used per "page." Icons can only be in one of three sizes in Signs and Cards, only one size in Banners, and can't be used at all in Screen Magic. Icons can only be placed in certain positions, too, only two relative sizes of text are possible on Signs, Cards, and Screen Magic, and only one on Banners. Each font is a different size, making font selection a matter of how much text you can fit on a page with it as much as how it looks. Banners can only be horizontally oriented, and borders are allowed on Signs and Cards only, and then only one one size and all the way around.

Of Course, there are ways around some of these restrictions, but using these ""workarounds" can become tedious at best. Multiple fonts and icons can be put on a "page" by making two or more pages, printed over each other. This, however, requires very careful design planning and paper handling. More font sizes can be obtained by using the Companion's font editor to create new fonts in new sizes other than those already existing.

Print Shop comes with several printer drivers, but noticeable absent are drivers for 24-pin printers and older models of some makes. 24-pin printers that will respond to 9-pin commands will work fine, but the extra graphics capabilities of these printers cannot be used with existing drivers. At one time, I attempted to use Print Shop with a Star Gemini

MAKING SIGNS WITH YOUR 8-BIT cont.

10, the precursor to the Gemini 10X. There is no Gemini 10 driver on my disk, and the 10X driver wouldn't work. Unfortunately, there is no provision for creating a custom driver. However, I have seen a PD driver for the Atari XMM801 advertised that purports to work with Print Shop, so someone must have figured out how to do it. Print Shop is written for one-drive systems, and there is frequent disk-flipping and swapping, another disadvantage. After totalling up the advantages and disadvantages, it was obvious that Print Shop, by itself, wasn't sufficient for our sign making needs.

NOTE FROM TWAUG.

There were two driver disks released by SECTOR ONE COMPUTERS (c) 1989. The contact number I have for them is, 313-978-2208.

One disk contains 2 24-pin Epson compatible drivers for Print Shop and Print Shop Companion, the other disk contains drivers for the same two programs, but for use with the Atari 1020 printer plotter.

Both Max and I have used the 24-pin drivers for Print Shop and the Companion, and the results were very good indeed.

SO, WHAT ABOUT PRINT POWER?

Print Power, by Hi Tech Expressions, a devision of Hi Tech Creations, is somewhat similar to Print Shop in scope, but there are several important differences. Some make it better, others make it worse.

The biggest advantage Print Power is its versatility. It, like Print Shop, has different output forms that can be pressed in to sign service-Signs, Cards and Banners. Unlike Print Shop, however, Print Power have multiple sizes and orientations available.

Print Power Signs may be oriented vertically (like Print Shop) or horizontally, and a half-page size is available. Cards may be tall (like Print Shop again), wide or "tent" style. Only horizontal orientations are allowed for Banners.

The versatility doesn't stop there. Up to four fonts may be used on one "page," and there are seven special effects available for each font, compared to Print Shop's three. Characters may be in three different sizes on Signs and Cards, and up to eleven on Banners. Banners may have multiple text lines. Print Power, and another Hi Tech program, Sesame Street Print Kit, are the only 8-bit programs I've ever seen that will produce multiple-line Banners.

There can only be one graphics per "page," but there are five different graphic sizes available on Signs and Cards. Three of these are automatically adjusted to fit all, half or a quarter of the area, while the other two are in two set sizes. You can place as many copies as you want of one size of a single graphic anywhere you want, as long as they don't overlap. Banner graphics can be placed on either or both ends of the Banner, and size is automatically adjusted to fit the text area. Banner graphics can be flipped on one end for a "bookend" effect.

Borders can be placed all around Signs and Cards, and also on the right, left, top, bottom, both sides, or top and bottom and can come in two sizes. Borders can also be used on the top, bottom, or both on Banners.

Of course, Print Power isn't perfect, it's biggest disadvantage is the lack of graphic editors. This means that you are locked into using the fonts, graphics, and borders supplied with the program and can only get others if Hi Tech decides a diskfull. At the time Print Power was released there was an additional graphics disk available, and Print Power does have the ability to utilize the graphics disk from another Hi Tech program, Awardware. Even with these disks, the selection of graphics pales beside that of Print Shop. No Frills Software once published a program called The Converter, which was supposed to convert Print Shop icons

MAKING SIGNS WITH YOUR 8-BIT cont.

into Print Power format, Borders and Fonts were not converted. I haven't seen that program in action, and have no idea of how well it works.

Print Power Requires a drive capable of using Atari enhanced density (DOS 2.5) disks, something which mistifies me. Back in 1987 or so, when Print Power was first marketed, there was, and still is as far as that goes, a good-sized percentage of 8-bit users with only single or double density drives, from various manufacturers. some of these drives could read an enhanced density disk, but couldn't write to it. Here's a case in point:

Print Power contains an undocumented features where the parameters of the last Sign, Card or Banner are saved to the disk in drive #1 so that they can be recalled and edited even after the computer is powered down. This is a handy feature, but if the disk in drive #1 has no empty sectors in the single density range, the program bombs. This happened repeatedly to a fellow ACE of syracuse member trying to use an Indus drive. A 1050 worked fine. Why would any commercial software developer not bent on self-destruction exclude a large portion of his potential market like that? It doesn't make sense to me.

Print Power supports several printers, including some 24-pin models. I did have a little problem at first, though. While the reference card inside the box stated that there was a driver for Gemini 10X, There was none on the disk. I used the one for the Star SG-10, the successor to the 10X, and that seemed to work, at least most of the time.

Print Power is also much slower than Print Shop, largely caused by its higher versatility. Some speed can be saved by using more than one drive putting the program disk in drive #1 and the graphics disk in drive #2. If you use an XL or an XE computer with more memory than a 64K 800, there is a way to increase that speed some more. Print Power has a memory reconfiguration option in its setup menu that allows the user to select what part of memory to use. This option supports expanded RAM machines, and is undocumented except for a small mote on the reference card under "Helpful Hints." print Power bit-maps its output in sections, filling a buffer and dumping it to the printer before going on to the next one. With a 48K configuration, a Banner saying "TOMATOES" in 3-inch letters took 10 "passes." Using 256K, that was reduced to only 2.

The memory reconfiguration resulted in the only problem I had with the Gemini 10X, a consequence, I believe, of not having the correct driver. At random times during printing the 10X would act as if it had dropped some bytes during transmission, getting hopelessly lost. The slower print speed of the 48K configuration seemed to compensate for the problem. So far, the Epson LX800, has performed flawlessly with Print Power, in all configurations.

Print Power does have an option in the setup menu that appears to be a custom printer/interface driver editor. Unfortunately, there is no documentation anywhere for it, and to navigate through it without some would be difficult indeed. I left it alone.

That automatic save option means that you can't have a write-protect tab on the disk. fortunately, Print Power is not copy-protected, and the documentation even recommends making a backup copy, so I made two backup copies. I now use only them and have the master disks carefully put away for safety.

Since acquiring Print Power, I use it more for signs than I do Print Shop. None of the graphics supplied are appropriate vegetable stand, so my Print Power signs are text-only. Although I might wish for a graphic now and then, it's much more important to have several sign sizes and shapes to choose from. The slowness, while annoying, is something I can live with.

MAKING SIGNS WITH YOUR 8-BIT cont.

AND SIGNMAKER

Signmaker is a shareware program written by Jeff Colehour in 1987. The latest incarnation, version 1.3, is available on Genie, (file 33a3), CIS (library 4, SGNMKR.ARC), and various other PD software sources. Older versions are floating around, too, but be sure to get version 1.3. It has the most features. Signmaker requires an XL/XE, a drive and a Star SG-10 compatible (in the two graphic densities used) printer. That would include most Epson-compatibles. It is written in compiled Turbo BASIC, so it needs a DOS compatible with that language. I use DOS 2.5 and occasionally, MYDOS. A large RAMDISK, while not essential, is certainly helpful. I use Tim Patrick's SmartRAM 2.5 from the July 1989 Antic.

Signmaker is designed to make 8 1/2 x 11 inch signs and flyers. It will print one copy each of two Print Shop icons, in one of four sizes, and uses up to three Atari graphics 0 fonts per page, in any of four sizes.

Signmaker Signs are laid out using a reference grid of 100 lines by either 60 or 120 columns, depending on the print density used. Icons may be placed anywhere on the page, using coordinates based on this grid. They may overlap each other, but do not merge with each other or with text. If overlap occurs, icon two will appear on top, followed by icon one, with text on the bottom.

Signmaker is particularly well-suited for creating detailed signs and flyers, and for the smallest signs. Its smallest text size allows up to 100 lines of 120 characters each, a lot of text for a sign. as with Print Shop and Print Power, many of the restrictions of the program can be overcome by careful planning and repeated printings on the same page. Signmaker's grid structure makes the planning stage much easier than for the other two programs.

Version 1.3 has three features not present in the original version that also aren't available with either Print Shop or Print Power. One of these is a special "double pass" print mode, analogous to a printer's double-strike printing. This not only produces a darker image with a worn ribbon, it also iliminates the thin black spaces that you sometimes get between print lines caused by uneven paper feeding.

Another feature is a "print preview" mode. This allows you to get a good idea of what your sign is going to look like without actually printing it out. Print Shop has no such feature, and while Print Power has a quicker "draft" quality print mode, you still have to print it out to see it.

The third feature is the ability to load text from a word processor file. This is especially useful for signs with large amounts of text, as the data imput and editor structures of Signmaker are rather primitive and difficult to use. Word processor editors are much easier to work with.

Signmaker can save the sign data in an ATASCII file. In fact, this is the preferred mode, as it is the only one that allows editing. This data file is easily accessible by word processors, and once you get used to the structure, it's easier to edit the file that way than it is with the program editor.

We use Signmaker for the smallest signs, the kind you put on individual boxes. At least, we use it when we're not in so much of a hurry that we hand-letter the signs with a felt-tip marker. We've also used it to make for the inserts for those clear plastic name tags. We don't use it as much as the other two programs, though, as it takes a lot more pre-planning. Also, the largest Signmaking text size are smaller and less readable than many Print Shop and Print Power fonts.

I HAVE A DREAM...

The program that would be best-suited for our signmaking doesn't exists, as far as I know, but I do know what I'd like to see in it. I'd like to combine the features of Print Shop, Print Power, and Signmaker, leaving out the things I

MAKING SIGNS cont.

don't like, of course. But more than that, I'd like to see those features enhanced.

I'd like to have the ability to specify the size of both text and graphics in an absolute measurement, like inches, centimeters, or whatever. I'd also like to see many sizes within the allowable range, so I can pick the right one instead of making do. I'd like to have absolute power over placement, including the size of the leading space "the white space between lines of text). I'd even like to angle my text once in a while.

Every 8-bit program I've ever seen that expands text or graphics does so by simply expanding the size of the pixels. In the larger sizes this method produces very jagged lines where smooth curves ought to be. I'd like to print large text and graphics without "jaggies."

I'd like to see drivers that take full advantage of the widest possible range of printers old and new, 9-pin, 24-pin, inkjet, and even laser. I might own one of these one day, and I'd like to be able to use it.

All this is a tall order for an 8-bit system, but I believe it's possible if the right programmer works on it. There are those who would say, "If you want to do all this stuff, why not move on to a different platform that already CAN do it?" Well, That's not an option for me, at least not for now. You see, most of all, I want to do it with an 8-bit.

XF551 ENHANCEMENT

Review by David Bryant (GKAUG)

How many of you have taken your trusty double sided floppy disk, punched a write protect hole in the jacket, and flipped it over to write on the reverse side? If you're like me, you've done it several times. So I thought nothing about it when I got my XF551 disk drive. I just put the disk in and loaded those basic programs like nothing was different.

XF551 ENHANCEMENT cont.

I was wrong. The first time I went to format a disk, (using Atari DOS 2.5) the drive protested and gave an error #173. I also had problems trying to write to the back side of disk in the XF551. I then got a gift from a friend, a modification for the XF551 that allows it to operate like my trusty 1050. Now formatting or writing to the back side of disks is no problem. This handy mod is called The XF551 Enhancement, from Computer Software Services.

If you look at a floppy disk, you will see a smaller hole near the edge of the large hole in the center. This is the index hole and some drives use it to tell when it is at the beginning of a track. The 1050 does not, so when you flip the disk, it doesn't matter if there is no hole on the reverse side. The 1050 uses a signal from the RAM I/O TIMER chip to provide the index pulse to the Floppy Disk Controller chip.

The XF551 uses a index hole sensor to provide the index pulse, and that's where this kit is needed! After you install this kit, a switch on the back of the drive allows you to choose XF551 or 1050 mode. In one position, the drive acts like a standard XF551. In the other position, you can format and write to the back side of disk like the 1050. Also you don't have to punch write protect holes in the jacket anymore.

The kit consist of an epoxy shell with the circuitry inside and a switch. About seven wires need to be attached to the circuit board and write protect sensor inside the XF551. Instructions included with the kit are very easy to follow. I would however remove the circuit board from the drive (only three more screws) when soldering to the IP jumper location. I located the switch above the Drive Select Switches and the epoxy shell next to it inside the rear cover. There's plenty of room and everything goes together easily.

I wish to thank Computer Software Services for a good product. Current cost is $29.95 and they will

XF551 ENHANCEMENT cont.

provide installation to those without soldering skills. They may be contacted at:

Computer Software Services
465 Kilbourn Rd.
Rochester, NY. 14618
Phone: (716) 467-9326
or

P.O. Box 17660
Rochester, NY. 14617

David Bryant is a devoted ATARI user, treasurer of G.K.A.U.G., and may be contacted at the clubs BBS (Phone: 616-657-2665).

NOTICE: This article originally appeared in the February, 1989 issue of Atari Interface Magazine and may be freely distributed or reprinted in non-profit User Group publications as long as the article's author and Atari Interface Magazine are credited AND this notice is reprinted with the article. All other publications must obtain written permission from Unicorn Publications, 3487 Braeburn Circle, Ann Arbor, MI 48108, Phone: (313) 973-8825 before using this article.

XIO EXPLAINED

Written by Paul Hollins

When you first bought your Atari, like me, you were probably dissapointed by the manual - or apparent lack of one! Who wasn't? I'd never used a computer before and with Atari neglecting vital pieces of information, it wasn't easy making those tentative first steps into programming. However, it did give me that little push that I needed to explore things on my own, without knowing what was going to happen. I think, therefore, it made it a lot more fun to learn this way than it would have otherwise been.

Recently when I was tinkering away at my keyboard, I thought, for some strange reason, about the elusive XIO command and what it was for - don't ask me why! I remembered that back in the days of the 'Atari Helpline' I had called them up to ask about it, as a fellow 8-bitter had told me what it was capable of. They said, "Oh yes, the XIO command. Well, err it's where you can, err, do things from BASIC. Err, that's it really!" Great, thanks for the help Atari! So, I set out trying to find out as much about it as I could on my own. I found a couple of fleeting references to it, but nothing spectacular. One article I remember reading called it "The Most Undocumented Command" and they weren't far wrong! My article has drawn inspiration (and in parts, information) from Ron Levy, who wrote the best description of using XIO's, in my opinion, in the now defunct MONITOR magazine, many moons ago!

XIO EXPLAINED cont.

WHAT IS XIO?

Well, BASIC has a number of commands for controlling data input and output, such as OPEN, PRINT, GET, PUT, etc. but obviously trying to squeeze everything that's needed in a BASIC language onto an 8K ROM, just didn't work. Atari ran out of space and consequently didn't have enough room to fit in all that they would have liked. Therefore, they created the XIO command as a means of being able to access the functions they didn't have room to name properly.

USING XIO

To use an XIO you have to adhere to the following rules, which are:

XIO, COMMAND No, #CHANNEL No, VALUE 1, VALUE 2, FILENAME$

The values associated with your XIO command will vary depending upon which function you would like to perform. Here's a brief explanation of what does what, when you issue an XIO command.

COMMAND No: The number you use here represents which command is going to be performed.

#CHANNEL No: Use this in the same way as you would in a normal OPEN command (i.e. #1, #2, #3, etc.).

VALUE 1/VALUE 2: These two integers (range 0 to 65535) are not actually

XIO EXPLAINED continued

used in many of the commands but a few XIO calls use them to pass parameters onto the operating system.

FILENAME$: Use this as a text area to pass text data such as a filename or that contained in a BASIC string.

SO WHAT CAN XIO DO?

The number of commands available is in fact down to you. The operating system of your computer was designed to allow for expansion and if you know how to program in machine code you can actually add more commands to the XIO table and those commands will be accessible from BASIC. But if you can't be bothered with all that and want to know what's available 'off the shelf' take a look at Fig One. As you can see, a lot of the commands are just duplicates of those already available from BASIC, but there are also several 'new' commands which aren't readily available. Let's take a look now at each one in more detail.

XIO 3 - OPEN A FILE OR DEVICE

The XIO version of BASIC's OPEN command is almost identical and, in practice, all your doing is replacing the command OPEN with XIO 3 to get the same result. So, next time you need to OPEN a device, you have a choice of two ways of doing it. The format of using this XIO is:

XIO 3,#1,8,0,"D:filename.ext"

As with the OPEN command, the 8 in VALUE 1 signifies 'open the file for output'.

XIO 9 - PUT RECORD OR 'PRINT'

To send a string of text to a file (which must have already been OPENed) use the following format:

XIO 9,#1,8,0,"HERE'S A LINE"

Note that you must place an 8 in VALUE 1 of the command, to signify an output. By the way, instead of using a literal string (in quotes), you can also print the contents of a variable to the file, for example:

XIO 9,#1,8,0,STRING$

XIO 12 - CLOSE FILE

Any CLOSE command you normally issue from within a program or in immediate mode (i.e. directly onto the screen without a line number) can be substituted with:

XIO 12,#1,0,0,"D:filename.ext"

This has exactly the same effect as BASIC's CLOSE statement.

READING A FILE

You can use an XIO command to read a file, which although has no advantage over using GET from BASIC does give you a better picture of what XIO is actually capable of. To use this feature, firstly you need to open the file to read. Instead of using BASIC's OPEN command use an XIO 3, but this time in VALUE 1 you should use a 4 to signify an input. For example:

XIO 3,#1,4,0,"D:filename.ext"

XIO 5 - GET RECORD OR 'INPUT'

To use an XIO 5 command in your program, you have to format it in the following manner:

XIO 5,#1,4,0,STRING$

And you must store the read from the file in a string. Of course this means you should DIM the string that your going to use, before you attempt to read the file otherwise you'll run into all sorts of problems. Also never try to read a record that is longer than the DIMmed length of your receiving string. Because if you do you may find that it corrupts other strings or even parts of your program, forcing BASIC to "lock-up"! And don't forget to close the file, after you've finished reading from it, by using an XIO 12 command.

THE MORE USEFUL COMMANDS

Okay, so we've taken a look at the XIO's that resemble BASIC

XIO EXPLAINED continued

commands, Now let's cast our beady eye over some others which aren't similar to any that are available from BASIC.

XIO 32 - RENAME FILE

By issuing an XIO 32, you can change the name of a file just as you would by using DOS. To use simply type:

XIO 32,#1,0,0,"D:oldname,newname"

And the result would be that file called 'oldname' would be renamed as 'newname'. All that without a DOS call in sight!

XIO 35 - LOCK FILE

This command takes care of the files on your disk, by allowing you to lock them all, or just one specific program or file. To lock just the one file type:

XIO 35,#1,0,0,"D:filename.ext"

And to lock all the files, you should type this instead:

XIO 35,#1,0,0,"D:*.*"

XIO 36 - UNLOCK FILE

If you need to unlock a file or files for any reason, it's just as simple as locking them. For example to unlock just one file type:

XIO 36,#1,0,0,"D:filename.ext"

And to unlock them all, type:

XIO 36,#1,0,0,"D:*.*"

XIO 254 - FORMAT DISK

This can be a real life saver, especially if you don't have a RAMDISK (or MEM.SAV) and you find that you have run out of disk space and need to format a new one so that you can save the program that you are writing. However if you do use this command, BE CAREFUL - because once you've issued it and pressed [RETURN] there's no turning back, everything on the disk will be erased as there are no 'Are You Sure?' prompts.

If you use DOS 2.5 you can in fact format your disk in either single or enhanced density. Firstly to format in single density use XIO 253 instead of XIO 254, for example:

XIO 253,#1,0,0,"D:"

But if you want to get as much disk space as possible you can format in enhanced density (on a 1050 or XF551) by typing:

XIO 254,#1,0,0,"D:"

Once you've done that, you can also write a DOS.SYS file to the disk by just typing SAVE "D:DOS.SYS". There will, however, be no DUP.SYS file - you'll still have to use DOS option H to do this. Incidentally, if your using an 810 you can issue either of the above commands, XIO 253 or XIO 254, to format your disk, but you'll only be able to get single density, I'm afraid.

BEWARE

All of the last five XIO commands that we've looked at must each be given an UNUSED channel. We have been using #1, but if #1 is already in use in your program then use a different one for any XIO commands. Whatever you do make sure that they don't clash or you'll get some strange error numbers.

XIO 18 - FILL

The last XIO is one that may come in handy for graphics programming. An example of the format to use is:

XIO 18,#1,0,0,"S:"

The fill function is used to fill in shapes which have been drawn on a graphics screen using the PLOT and DRAWTO commands. It is, however, very limited in the way that it operates is very simple, in fact it is probably quite a big disappointment to most people when they learn how to use it for the first time!
Type in and run Program One. Notice that it draws a line on the right hand side of the screen, and then proceeds to fill it.

XIO EXPLAINED continued

HOW DOES IT WORK?

The process the computer follows when executing a fill is very simple indeed. It goes as follows:

1. Draw an imaginary line between the pixel last referenced to in a PLOT or DRAWTO, and the pixel referred to in the POSITION command.
2. Go to the beginning of this line, and for each pixel along the imaginary line, draw a horizontal line out to the right, until it meets another lit (non-zero) pixel. If it does not meet another lit pixel, it will 'wrap around' the screen until it meets itself!

Following this logic, it should be easy to see that in order to draw a square, or filled-in box, you need only draw the right-hand edge, then give the top left and bottom left corner's coordinates in the PLOT and POSITION commands, and the fill will then take care of the rest. Try experimenting with different shapes and you'll soon begin to see how limited, in practice, Atari's fill really is!

COMMAND	OPERATION	EXAMPLE
3	Open a file	Same as BASIC's OPEN
5	Get record	Same as BASIC's INPUT
7	Get characters	Same as BASIC's GET
9	Put record	Same as BASIC's PRINT
11	Put characters	Same as BASIC's PUT
12	Close a file	Same as BASIC's CLOSE
13	Status request	Same as BASIC's STATUS
17	Draw a line	Same as BASIC's DRAWTO
18	Fill an area	XIO 18,#1,0,0,"S:"
32	Rename a file	XIO 32,#1,0,0,"D:oldname,newname"
33	Delete a file	XIO 33,#1,0,0,"D:filename.ext"
35	Lock a file	XIO 35,#1,0,0,"D:filename.ext"
36	Unlock a file	XIO 36,#1,0,0,"D:filename.ext"
37	Point	Same as BASIC's POINT
38	Note	Same as BASIC's NOTE
253	Format Single	XIO 253,#1,0,0,"D:"
254	Format enhanced	XIO 254,#1,0,0,"D:"

Fig One

```
 0 REM ┌──────────────────────────┐
 1 REM │        PROGRAM 1         │
 2 REM ├──────────────────────────┤
 3 REM │A Demonstration of using  │
 4 REM │      XIO 18 (FILL)       │
 5 REM │   Written by Ron Levy    │
 6 REM ├──────────────────────────┤
 7 REM │This program originally   │
 8 REM │appeared in MONITOR 16    │
 9 REM └──────────────────────────┘
10 GRAPHICS 8
20 COLOR 1
30 REM ** DRAW RIGHT EDGE **
40 PLOT 300,20
50 DRAWTO 300,100
60 REM ** DEFINE LEFT EDGE **
70 PLOT 30,20
80 POSITION 100,100
90 REM ** GIVE FILL COLOUR **
100 POKE 765,1
110 REM ** CALL THE FILL **
120 XIO 18,#1,0,0,"S:"
```

CODING CAPERS

by Andrew C. Thompson.

Hello there again, welcome to another thrilling caper of coding. This issue I'm going to do something really technical, mainly aimed at all you real heavy coding buffs! Yeap, this issue I'm going to show you how to perform a sine-wave with text!

This particular text sine-wave is of standard format and is in a direct mode of execution. What I mean by this is that the routine completes its purpose in 1 loop, it doesn't escape the loop processing small bits at a time. OK now, let's chuck you the code and we'll have a little yat about it later. By the way, you'll find the code is pre-assembled into particular memory for you, if you don't want the given memory then you can check the variable table later for the info to alter it:

```
CharData locator:

0730  LDA  $0800,X
0733  STA  $077E
0736  LDA #$00
0738  STA  $077F
073B  ASL  $077E
073E  ROL  $077F
0741  ASL  $077E
0744  ROL  $077F
0747  ASL  $077E
074A  ROL  $077F
074D  CLC
074E  LDA  $077F
0751  ADC #$E0
0753  STA  $077F
0756  RTS

Rotate VertPos's:

07BB  LDA #$00
07BD  STA  $0731
07C0  LDA  $07D8
07C3  PHA
07C4  LDX #$00
07C6  LDA  $07D9,X
07C9  STA  $07D8,X
07CC  INX
07CD  CPX #$28
07CF  BNE  $07C6
07D1  PLA
07D2  STA  $07FF
07D5  JMP  $07A1

MainRoutine: Poke the data on-screen:

0757  JSR  $0730
075A  LDA  $070C
075D  STA  $0781
0760  LDA  $070D
0763  STA  $0782
0766  LDA  $07D8,X
0769  TAY
076A  CLC
076B  LDA  $0781
076E  ADC #$28
0770  STA  $0781
0773  BCC  $0778
0775  INC  $0782
0778  DEY
0779  BPL  $076A
077B  LDY #$00
077D  LDA  $E000,Y

0780  STA  $0880,X
0783  CLC
0784  LDA  $0781
0787  ADC #$28
0789  STA  $0781
078C  BCC  $791
078E  INC  $782
0791  INY
0792  CPY #$08
0794  BNE  $77D
0796  NOP
0797  CLC
0798  INC  $0731
079B  BCC  $07A0
079D  INC  $0732
07A0  RTS

Do all Chars:

07A1  LDX #$08
07A3  JSR  $0757

Initiation:

07AE  LDA #$08
07B0  STA  $0230
```

CODING CAPERS continued

```
07A6  INX              07B3  LDA #$07
07A7  CPX #$28         07B5  STA  $0231
07A9  BNE  $07A3       07B8  JMP  $07A1
07AB  JMP  $07BB
```

There's the code. The Display List (DL) in this assembly begins at $0708 and ends at $072F. It's just Graphics-$8 mode-lines, the DM pointer by the way begins at $1000.

The variable table for this assembly is:

Addr:	Explanation:
$070C,D	DM backup addr., also the actual addr. within the DL,
$077E,F	Font and Char address,
$0781,2	DM + Vert. position offset,
$07D8,X	40 Vert. position offsets obtained with the X-Index,
$0800,X	This is your 40-byte text.

The basic process of the routine runs on a few simple steps:

1. Setup the DL and DM,
2. Locate and point to present char.,
3. Calculate DM of present char. and add the appropriate vert. offset,
4. Poke the char. onto the screen into the pre-calculated address,
5. Re-loop to step-2 until all the 40 chars. are displayed,
6. Once all done, rotate the vertical position offsets to cause motion,
7. Now, just loop forever back to 2.

If you've never tried anything like this before then you will find it a little bit tedious but give it a little patience and it'll click. Anyway, although this does the job of sine-waving text there still are many more different and better ways to achieving this. For example; A couple of ways of speeding up the loop is to alter and control the instruction at address $0766 to access the vertical position table so that you don't have to waste crucial time rotating them with the routine beginning at $07BB. Another way is to reduce the characterset to 5-rows per char. and limit the charset to 50 chars, this way you can access the entire 5-rows data of the character with an index register instead of having to load the LSB of the address, adding 40, storing it back and altering the high-byte also! It might not sound very much, but when the difference is put in a loop of 40 characters going to the screen such as this one, your talking of saving a couple of thousand processing cycles!! Also, the mode does not have to be like it is, you could if you liked process 20 chars per Vertical-Blank or whatever.

Anyway, I hope I've given some of you an insight into how such a routine would be accomplished.

CODING CAPERS continued

There are far more complex ones than this one, especially when you nest the code of one routine with many others, things usually tend to become BUGGED! In actual fact I've actually just recently finished what you might call a "Spiral-Scroller". I'll be putting it together into a DEMO sometime when I get around to it. In short, the routine I've just done will convert any circular drawn pattern into character plotting addresses and swing a given amount of characters through 2560 XY co-ordinates over 4K of Graphics mode #8. When I was creating this scroll I came across a problem of processing time, since the whole routine to plot the characters onto screen takes around 100000 cycles, and since you only get a 10th of this time per frame I had to achieve a way to do this without any visual processing alterations. The technique to do this is a relatively simple one and it is a trick with the Display Memory, and I believe it is actually used in a lot of high quality Polish DEMO's also. Take the 24-scroll DEMO by MAGNUS on the TOP DEMO disk for example. How do you think he achieved 24 sine-wave scrolls at the same time!? Well, I'll leave you in ponderment about this technique now, in the meantime I'll try to get this routine into a DEMO for all to see. See you next time!

S.A.M.S

TWAUG would like to thank everybody who came to our stand to say hello and supported the 8-bit, on Saturday the 15th of April at the Spring All Micro Show. We also like to give a special thanks to Collin Doyle, Daniel Orts and Allan Turnbull who gave us a helping hand on our stand.

The show wasn't all that well supported this time, we believe it was due to being the Easter weekend. There were a number of empty stands in the hall, Micro Discount was missing too, we knew that Derek Fern had other arrangements booked for that holiday weekend and we are sure other regular visitors had made different arrangements too.

We, the TWAUG people enjoyed the show but then we always do. We are happy just to be there and show our faces the show gives us the opportunity to advertise our newsletter. WE also like to say hello and have a chat with our friendly subscribers who live in different parts of the country. We got a few new subscribers this time around and we also sold a few 8-bit items. All in all we came away even, as we are a non-profit making User Group we never worry about making any profit.

We will be attending the show in November again, our stand has been booked and also our hotel rooms. Do you think we are enthusiastic about the show???

LACE

The London Atari Computer Enthusiasts.

As a member of LACE you receive a monthly newsletter and have access to a monthly meeting. They also support the ST with a good selection of PD sotware.

The membership is £8.00 annually.

Write to the secretary of LACE for more information:

Mr.Roger Lacey
LACE Secretary
41 Henryson Road
Crofton Park
London SE4 1HL
or phone: 0181-690 2548

CRACKING THE CODE

by Keith Mayhew

Re-printed by M. Gerum

This article first appeared in "The UK ATARI Computer Owners Club" later renamed "MONITOR"

Part 15

In the last part of this series we looked at how the CIO allows an independent and simple method for transferring data to or from devices. This is the level at which most programs would work for handling keyboard input, screen output, disk files etc. This time we look at how the CIO actually implements the commands which are given to it and look at how to implement a new device handler and install it,

HOW CIO EXECUTES COMMANDS

Figure 1 illustrates the structure of the parts of the O.S. which are involved in input/output operations and also labels the main data structures used to communicate between routines.

The main interface to the O.S. from the user program is the CIO which communicates with one of the eight IOCBs at any one time. In performing the specified operation the CIO copies the revelant IOCB into an internal structure in page zero called ZIOCB at 20hex. It then determines where the handler for the specified device resides and uses it to perform the actual input or output for that device. CIO generally modifies the ZIOCB as the operation progresses and copies the contents of the ZIOCB back into the original IOCB upon completion of the command.

As not all device handlers are resident in the system, most notably the disk drive and the RS232 interface, the O.S. employs a table to keep track of the installed handlers. This method makes it simple not only to add new device handlers at any time but also to modify or replace the operations of existing handlers.

The device handler table HATABS at 31A hex, has room for a maximum of twelve entries, each consisting of 3 bytes. Initially the system fills in the table with its resident handlers and sets spare entries to zero. Each entry consists of the letter of the device in upper case ASCII followed by a two byte address, stored in usual reverse order, which points to the handler's vector table.

When CIO is requested to open an IOCB it searches the handler table for an entry with a matching device name. This search is performed from the end of the table so that if there is a multiple entry for a device, the one nearer the end of the table is chosen. CIO places this index into HITABS into the user's IOCB at ICHID. If the user specified a device number, such as D2:, then CIO places this number into ICDNO. If no number is specified then CIO assumes one, thus D: is the same as D1: to the CIO. Having recorded this information from an open command means that further calls are then much faster. When an IOCB is closed its ICHID value is set to FF to indicate it is free.

DEVICE HANDLERS

Each device handler as mentioned above, has a vector table. This table consists of a list of the entry points to the handler's code as illustrated in figure 2. What might find rather confusing that each address of a routine in the table is one less than the actual address. This is done because 6502 does not have a JMP indexed-indirect instruction. To make the operation as fast as possible the CIO uses a little "trick," it pushes the two bytes onto the stack and executes an RTS instruction. RTS expects to find the address of the jump location minus one on the stack as this is what its counterpart, JSR, puts onto the stack as a return address.

Each handler has access to the zero page IOCB, ZIOCB, its entries have the same meaning as the originating IOCB but they do not necessarily hold exactly the same values. The locations and their labels are as listed below;

20 = ICHIDZ = Handler index.

CRACKING THE CODE continued

21 = ICDNOZ = Device number.
22 = ICCOMZ = Command.
23 = ICSTAZ = Status.
24 = ICBALZ = Buffer address low.
25 = ICBAHZ = Buffer address high.
26 = ICPTLZ = Put-byte routine low.
27 = ICPTHZ = Put-byte routine high.
28 = ICBLLZ = Buffer length low.
29 = ICBLHZ = Buffer lenght high.
2A = ICAX1Z = Auxiliary byte 1.
2B = ICAX2Z = Auxiliary byte 2.

Note that the extra 4 auxiliary bytes bytes in the original IOCB are not copied into or out og the ZIOCB as these locations are used for other purposes internally to CIO. If a handler needs access to these values it must go to the original IOCB.

When a handler routine is called by the CIO, the X register contains the IOCB index the user originally supplied so that it is possible to access the originating IOCB. The Y register is automatically set to 92hex which is the error code for function not implemented, thus if a routine is not used by a handler it need only perform an RTS. If the handler does provide a routine for any functions then it must set the Y register to 1 for success or a suitable error code before returning. Following are the descriptions of the functions a handler can provide.

Handler OPEN

The handler has to prepare for IO with its device. ICBALZ and ICBAHZ point to the filename, which will be ignored unless the device supports named files, e.g. a disk. The device number is always available to the handler from ICDNOZ if it supports more than one device. The auxiliary byte ICAX1Z and ICAX2Z should be checked to make sure that they confirm with the device. For example, the handler should not allow writing to a read-only device.

Handler CLOSE

The handler should make sure that all pending operations are completed, such as writing data to a device, before returning whereupon CIO will mark the IOCB as free.

Handler GET BYTE

The handler should return the next byte from the device in the A register.

Handler PUT BYTE

The handler should put the byte in the A register out to the device.

Handler GET STATUS

The handler should respond by filling any necessary bytes into the four bytes starting at DVSTAT at 2EA hex. For most handlers this function will not be of any use as normal status of operations are returned via the Y register and subsequently stored in the IOCB by the CIO. If this is the case then the handler should always return one in the Y register indicating this routine succeeded. It will not have to alter the status bytes as they are device dependent.

Handler SPECIAL

If the CIO finds a command byte with a value of 0D hex or greater then it calls the SPECIAL entry of the handler with the contents of the ZIOCB the same as the originating IOCB. If the handler does not support any extended commands then it should just return. An example of a special command is the DRAWTO command of the display handler; the handler can determine which special command is requested by looking at ICCOMZ.

For SPECIAL commands or a GET STATUS command the CIO supports an implied open mode. This allows such a command to be performed on a closed IOCB. CIO first opens the IOCB, performs the action needed and then closes it before returning. The buffer address must point to a suitable device/filename and the auxiliary bytes must be set for a normal open command if the implied open method is to succeed.

Handler INITIALISATION

The last entry in a handler's

CRACKING THE CODE continued

vector table is a JMP instruction to code which is called for initialisation at reset and power-up. Unfortunately the person who wrote CIO forgot to call these vectors at all!! The initialisation should just return one in the Y register. The problem caused by this omission is that after a reset the handler table which keeps the device names is re-built, so any devices which were added are now removed and hence their names are unknown to the system. You can still get round this by patching into the vector for a reset and then re-installing the device name.

The BASIC Complication

The poeple who wrote ATARI BASIC managed to get a loophole put into the CIO to enable faster writing operations from BASIC, particularly for byte-at-time output to a device. This loophole is the put-byte vector in each IOCB: ICPTL & ICPTH, and their counterparts in the ZIOCB: ICPTLZ & ICPTHZ. The put-byte vector of the IOCB is a duplicate of the entry in the handler vector table.When an IOCB is opened, CIO copies the vector from the handler to the IOCB; when closed the entry is set to point to a routine which returns an error code for IOCB not-open.

The presence of this put-byte vector is usually of little significance but can cause unexpected results if you are not careful. The main thing is that you should not rely upon the values in the ZIOCB for a put-byte operation and look at the originating IOCB instead. The handler should also check that a write operation is being performed to the device if it was opened in read-only mode.

As the operating system does not use this put-byte vector you only have to consider the problems if the device will be used from within the BASIC environment.

OVERVIEW OF SERIAL INPUT OUTPUT

As can be seen from Figure 1, some handlers interface directly with the hardware whereas others use the sub-system known as the Serial input Output (SIO). It is the SIO which handles all transfers of data to external devices connected via the 13- pin serial port, such as disk drives, printer or cassette.

Communication between a handler and SIO is done via the Device Control Block (DCB) data structure. Usually you would not set up values in the DCB and call SIO directly but use the appropriate handler via a call to CIO and let the handler deal with the complications for that particular device.

The exceptions to this structure is the disk drive which has a resident disk handler which provides a very low level access to a disk on a sector basis. The user has direct access to this handler but its main purpose is to read and write sectors for the CIO non-resident device handler D:. This is what is refered to as the Disk Operating System (DOS) and provides a mutiple-file structure for a disk. To maintain these files DOS allocates some sectors of the disk for a directory of the names of the files and where yhey are actually stored on the disk.

A NEW DEVICE HANDLER

Putting theory into practice, we will now look at how a new device handler can be written. Listing 1 is the assembly code for a device handler called B: which provides a buffer in memory for data to be stored in and read out of. With B: you can copy a small file into it from DOS and then copy it onto, say, another disk. It is not a RAM disk in the sense that you cannot write more than one file to the device. However, a special feature of B: is that in fact it is a queue.

A queue allows you to write data to it and then, while the device is still open, start reading the data already written. If the data being written reaches the end of the buffer it can wrap around to the beginning if some data has already being read. In effect this means that an infinite amount of data can be written to the device as long as data is also read. This could be useful in a program which generates a lot of intermediate

CRACKING THE CODE continued

data and wants somewhere to write it before another part of the program reads the data again.

Listing 2 is a BASIC program which reads the device handler into page 6, it is initialised with X=USR(1536). The device uses 1K of memory starting at 6000 hex as the circular buffer.

The program starts by finding a spare entry in the handler table, HATABS, where it can insert the character "B" and the address of the vector table VECTAB: this installs the device in the CIO. There are four pointers which are initialised for the device, these are: BUFSTRT which points to the start of the buffer;
BUFLAST which points to the end of the buffer; BUFPUT which points to where the next character will be put; BUFGET points to where the next character will be read from. The number of bytes in the buffer is held in BUFLEN, this is initially zero.

The handler only has code for OPEN, GET & PUT. The rest of the operations are not needed and so are set to suitable return points in the program. The OPEN function performs no function if the auxiliary byte does not specify a write operation: this is because if there is data in the buffer, the pointers do not need to be changed. For an open with a write, the pointers are reset and any previous data is lost.

The GET function returns the next available byte assuming that BUFLEN is non-zero, i.e. there are bytes to be read. It decrements BUFLEN if a byte was read and moves BUFGET to the next character. If BUFLEN was zero then end-of-file status is returned.

The PUT function behaves similarly to GET except that it will only insert a byte if BUFLEN is not at its maximum value. If the buffer is full then the DOS error status disk-full is returned.

The get and put functions use the routines INCRMNT and DECRMNT to either add one or take one from a pointer. These routines take the address of pointer in the X register and access the bytes using 0,X. and 1,X. This makes the routines general purpose and saves duplicating code in the program for each pointer. The routine BUFWRAP also takes the address of a pointer in the X register and sets the pointer to the start of the buffer if it is currently past the end of the buffer.

Note that the PUT routine performs a test on the IOCB's auxiliary byte to ensure that BASIC is not performing a direct put call while the IOCB is only open for reading!

There are two problems with the way this device has been written. The first is that the buffer is in a place in memory that could easily be overwritten by another program such as BASIC. However, it is difficult to provide a completely safe area of memory! The second problem is that a system-reset will remove the device from the handler table and will need to be called again to re-install itself.

NEXT TIME

That is all for this time. In the next part we will examine the key features of each of the system handlers and also explain how the two problems mentioned above could possibly be solved.

Listing 2.

```
QZ 10 DIM HEX$(16)
CV 20 LINE=10000:TRAP 100:J=0:START=1536
VA 30 READ HEX$,CHKSUM:SUM=0
AA 40 FOR I=1 TO 15 STEP 2
ZG 50 D1=ASC(HEX$(I,I))-48:D2=ASC(HEX$(I
   +1,I+1))-48
KT 60 NUM=((D1-7*(D1>16))*16+(D2-7*(D2>1
   6)))
LW 70 SUM=SUM+NUM:POKE START+J,NUM:J=J+1
   :NEXT I
LY 80 IF SUM=CHKSUM THEN LINE=LINE-10:GO
   TO 30
IN 90 ? "Checksum error on this line:"
VO 95 LIST LINE:END
YS 100 PRINT "Data in memory."
YQ 10000 DATA 68A221BD1A03F006,763
NZ 10010 DATA CACACA10F660A942,1199
DI 10020 DATA 9D1A03A9D99D1B03,759
LY 10030 DATA A9069D1C03A9006D,673
```

CRACKING THE CODE continued

```
JI 10040 DATA EB0685CB85CDA9FF,1339
FF 10050 DATA 8DED06A90085CF85,1026
YL 10060 DATA D0A9608DEC0685CC,1193
FI 10070 DATA 85CE1869038DEE06,856
NG 10080 DATA 60A52A2908F012AD,783
RG 10090 DATA EB0685CB85CD85CF,1255
SY 10100 DATA 85D0ADEC0685CC85,1226
DO 10110 DATA CEA00160A5D0D004,1048
HW 10120 DATA A5CFF016A000B1C3,1174
ZR 10130 DATA 48A2CB20AA0620BC,865
BT 10140 DATA 06A2CF20B10668A0,854
MZ 10150 DATA 0160A08860A8BD4A,920
FR 10160 DATA 032908F01FA5D0C9,897
MQ 10170 DATA 04D004A5CFF01898,1004
UI 10180 DATA A00091CDA2CD20AA,1079
XH 10190 DATA 0620BC06A2CF20AA,303
XF 10200 DATA 06A00160A08760A0,314
IY 10210 DATA A260F600D002F601,961
FF 10220 DATA 60D600B500C9FFD0,1155
GS 10230 DATA 02D60160B501CDEE,938
EJ 10240 DATA 069015D009B500CD,774
LN 10250 DATA ED06900CF00AADEB,1057
PY 10260 DATA 069500ADEC069501,720
RR 10270 DATA 604006E7065B0675,624
TD 10280 DATA 06E706E9064CE806,796
ZP 10290 DATA 400160,257
```

Figure 1.

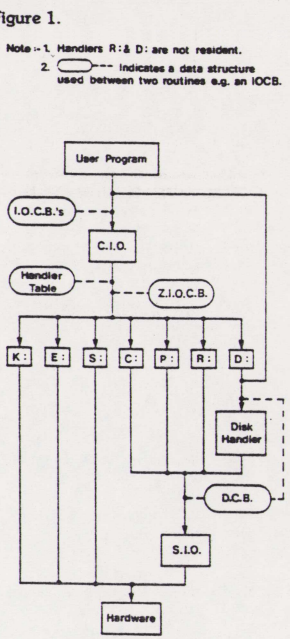
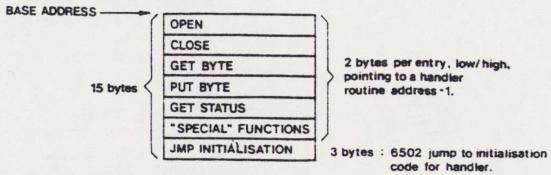


Figure 2.



Listing 1.

```
0100 ; Handler for a buffer device 8:
0110 ICAX1Z  =      $2A     ;Zero page auxiliary 1.
0120 HATABS  =      $031A   ;Handler table.
0130 ICAX1   =      $034A   ;IOCB auxiliary 1.
0140 BUFPGS  =      4       ;Number of pages for buffer.
0150 WRITE   =      8       ;Open for write bit.
0160 RDONLY  =      135     ;Error - read only.
0170 ENDFILE =      136     ;Error - end of file.
0180 DSKFULL =      162     ;Error - disk full.
0190         *=     $CB
0200 BUFGET  *=     *+2     ;Buffer get pointer.
0210 BUFPUT  *=     *+2     ;Buffer put pointer.
0220 BUFLEN  *=     *+2     ;Number of bytes in buffer.
0230 BUFADDR =      $6000   ;Address of buffer.
0240         *=     $0600
0250         PLA
0260         LDX    #33     ;Search HATABS from end.
0270 NEXTDEV LDA    HATABS,X
0280         BEQ    EMPTY   ;Spare slot?
0290         DEX
0300         DEX
0310         DEX
0320         BPL    NEXTDEV ;Next device entry.
0330         RTS            ;No room found!
0340 EMPTY   LDA    #'B'    ;Name of handler.
0350         STA    HATABS,X ;Point to vector table.
0360         LDA    #VECTAB&$FF
0370         STA    HATABS+1,X
0380         LDA    #VECTAB/256
0390         STA    HATABS+2,X
0400         LDA    #$00    ;Buffer address low.
0410         STA    BUFSTRT ;Start of buffer.
0420         STA    BUFGET  ;Get pointer.
0430         STA    BUFPUT  ;Put pointer.
0440         LDA    #$FF
0450         STA    BUFLAST ;End of buffer low.
0460         LDA    #0      ;Set length to zero.
0470         STA    BUFLEN
0480         STA    BUFLEN+1
0490         LDA    #BUFADDR/256 ;Buffer address high.
```

CRACKING THE CODE continued

```
0500         STA   BUFSTRT+1
0510         STA   BUFGET+1
0520         STA   BUFPUT+1
0530         CLC
0540         ADC   #BUFPGS-1 ;Add number of pages.
0550         STA   BUFLAST+1 ;End address high.
0560         RTS           ;Installed.
0570 ;Handler OPEN.
0580 OPEN    LDA   ICAX1Z  ;Get auxiliary byte.
0590         AND   #WRITE  ;See if write mode.
0600         BEQ   OPENOK  ;No.
0610         LDA   BUFSTRT
0620         STA   BUFGET  ;Reset get & put.
0630         STA   BUFPUT
0640         STA   BUFLEN  ;Reset length.
0650         STA   BUFLEN+1
0660         LDA   BUFSTRT+1 ;And high bytes.
0670         STA   BUFGET+1
0680         STA   BUFPUT+1
0690 OPENOK  LDY   #1
0700         RTS
0710 ;Handler GET.
0720 GET     LDA   BUFLEN+1 ;Any bytes in buffer?
0730         BNE   GETOK
0740         LDA   BUFLEN
0750         BEQ   EOF     ;No - end of file.
0760 GETOK   LDY   #0      ;Get byte.
0770         LDA   (BUFGET),Y
0780         PHA           ;Save it.
0790         LDX   #BUFGET ;Increment GET pointer.
0800         JSR   INCRMNT
0810         JSR   BUFWRAP ;Wrap around if necessary.
0820         LDX   #BUFLEN ;Decrement byte count.
0830         JSR   DECRMNT
0840         PLA           ;Restore byte.
0850         LDY   #1      ;Good status.
0860         RTS
0870 EOF     LDY   #ENDFILE ;End of file status.
0880         RTS
0890 ;Handler PUT.
0900 PUT     TAY           ;Save byte.
0910         LDA   ICAX1,X ;Make sure write allowed!
0920         AND   #WRITE
0930         BEQ   NOWRITE ;Read only.
0940         LDA   BUFLEN+1 ;Room in buffer?
0950         CMP   #BUFPGS
0960         BNE   PUTOK
0970         LDA   BUFLEN
0980         BEQ   FULL    ;No - buffer full.
0990 PUTOK   TYA           ;Restore byte.
1000         LDY   #0      ;Save it.
1010         STA   (BUFPUT),Y
1020         LDX   #BUFPUT ;Increment put pointer.
1030         JSR   INCRMNT
1040         JSR   BUFWRAP ;Wrap around if necessary.
1050         LDX   #BUFLEN ;Increment byte count.
1060         JSR   INCRMNT
1070         LDY   #1      ;Good status.
1080         RTS
1090 NOWRITE LDY   #RDONLY ;Read only status.
1100         RTS
1110 FULL    LDY   #DSKFULL ;"Disk" full status.
1120         RTS
1130 ;Increment pointer at X.
1140 INCRMNT INC   0,X     ;Low byte.
1150         BNE   INCRET
1160         INC   1,X     ;High byte.
1170 INCRET  RTS
1180 ;Decrement pointer at X.
1190 DECRMNT DEC   0,X     ;Low byte.
1200         LDA   0,X
1210         CMP   #$FF
1220         BNE   DECRET
1230         DEC   1,X     ;High byte.
1240 DECRET  RTS
1250 ;Wrap pointer to start of buffer.
1260 BUFWRAP LDA   1,X     ;High byte.
1270         CMP   BUFLAST+1
1280         BCC   NOWRAP  ;Less than.
1290         BNE   WRAP    ;Greater than.
1300         LDA   0,X     ;Low byte.
1310         CMP   BUFLAST
1320         BCC   NOWRAP  ;Less than.
1330         BEQ   NOWRAP  ;Equal to.
1340 WRAP    LDA   BUFSTRT ;Wrap to start of buffer.
1350         STA   0,X
1360         LDA   BUFSTRT+1
1370         STA   1,X
1380 NOWRAP  RTS
1390 ;Vector table for device.
1400 VECTAB  .WORD  OPEN-1  ;Open.
1410         .WORD  RETOK-1 ;Close.
1420         .WORD  GET-1   ;Get.
1430         .WORD  PUT-1   ;Put.
1440         .WORD  RETOK-1 ;Status.
1450         .WORD  RETURN-1 ;Special.
1460         JMP    RETOK   ;Initialization.
1470 RETOK   LDY    #1      ;Good status.
1480 RETURN  RTS
1490 BUFSTRT *=     *+2     ;Pointer to start of buffer.
1500 BUFLAST *=     *+2     ;Pointer to end of buffer.
```

ADVENTURES REVIEW

Unfortunately, Mark Stinson is unable to make his usual contribution due to the fact that at the moment, he is in the midst of moving house and changing jobs. Hopefully Mark will be back in the next issue of TWAUG, but for this issue, I have picked out a couple of game reviews from back issues of the Analog Magazine. I hope you like them.

THE DARK CRYSTAL

By Brian Moriarty.

First appeared in Analog issue 13.

Dark Crystal is an illustrated adventure based on the Jim Henson movie of the same name. The object is to locate and restore the missing shard of a magic crystal before the "Great Conjunction" of three suns. I haven't seen the film, so I had no prior knowledge of the story's plot or characters. My comments here are based solely on the inherent qualities of the game and not on its value as a souvenir.

The program occupies both sides of three disks. Disk 1, side A is the main interpreter; the other sides contain picture data for the dozens of colour illustrations. On-Line thoughtfully provides a backup utility that lets you copy the picture disks, which are subject to lots of wear and tear. The interpreter is copy-protected, however.

Game play is similar to On-Line's popular Wizard and the Princess, and Mission: Assault adventures. The parser is of the simple two-word, verb-nown variety; multiple commands or complex sentences are not allowed. Each game location has its own hi-res colour illustration which must be pulled off the disk. You can "flip away" the picture temporarily to view a listing of your last several commands. Provisions are made for the saving and loading of up to 15 different game positions. The program also lets you format an extra game-save disk during the course of play -- a lifesaver if you are in a tough spot with no formatted disks handy.

The illustrations for Dark Crystal are supposed to have been digitized from actual movie stills. Detail and colouration are still rather crude -- certainly not photographic -- but the selection includes a number of dramatic perspective and shading effects you don't offen see in games of this type.

I spent the better part of an evening mapping out Dark Crystal. I wandered through almost forty different locations and exhausted three of the five picture-disk sides. Aside from a few unavoidable encounters with characters telling you what to look for, nothing happened. There were no threatening situations, no puzzles, mazes or unusual objects to pick up, just cute little creatures peeking out from behind trees, and a couple of dead ends. The handsomely printed owner's guide tells you most of what you need to know about your mission; very little itself to the imagination.

Dark Crystal seems to be more concerned with recreating the events and scenery of the movie than with providing a fun game. Despite its fine packaging and professional engineering. I still prefer On-Line's previous Hi-Res adventures for the ATARI.

TEMPLE OF APSHAI TRILOGY

By Patrick J. Kelly.

It is dark in the ancient temple. Outside of your own rasping breath in your ears, the only sound is the faint rush of wind down a nearby chamber. Even though you've been through many such adventures, you cannot help but feel a tinge of apprehension. Your intuition, honed to a fine edge tells you that something is around the next bend. In this near darkness, your eyes are of little help to you, so you must rely on your finer senses.

Adjusting the grip of your small shield, you edge forward. The hefty

ADVENTURES.REVIEW continued

short-sword feels good now, its leather wrapping clenched in your sweaty palm. With a slight mental shrug, you curse yourself purchasing plate arrows; the chain mail covering your body will not block the sting of poison barbs or claws. As you edge forward, step by slow step, you hear the rustling of the unseen something. Here it comes -- confrontation.

Steeling your nerves, you whirl with the sword, ready to slash out at the now-visible form. Suddenly, out of the shadows, something lunges. It is, amazingly, a man-sized ant! No time for your amazement now; you raise your shield against the antman's pike. The battle for the Temple of Apshai has begun.

If the introduction to this review is a bit wordy, you must pardon me. However, as any of you who have ever played fantasy games know, words are the key to a successful game--or, if not words themselves, then the images thus cunjured up. In this arena of the mind, the more elaborate the image, the better.

In the Epyx compilation of the Apshai games, Temple of Apshai Trilogy, these images flow like water well-spring. This latter adventure takes the best of three games, tie them together with a central theme and lets you loose into the universe of Apshai.

In the first, you -- the handy freebooter -- must venture into the crumbling Temple of Apshai in search of gems and gold. During your exploration of the ruins, you must evade various traps, monsters, ghouls and thingamabobs, to survive and emerge victorious.

Still greedy? Don your armor again and explore the Upper Reaches of Apshai. This is where the fun realy starts, and the dangers you must face increase. In this round, you're wandering about the deserted citadel of the Apshaians, again looking for things to wear, eat and bank.

If you make it through these funfests, the Curse of Ra awaits you. Oooh! Wanna a hint? Curse of Ra takes you into the desert this time, wandering the parched sands for the ultimate goal of financial fulfillment (or, for the less mercenary among us), the satisfaction of solving the Curse of Ra.

If all of this sounds like your gobler of ale, come on in. The adventure of a lifetime awaits.

BEGINNINGS AND OTHER SHANANIGANS

You start your career as an adventurer in the inn. Choosing your character's name, strength, armory, healing prowess and intelligence. You can let the computer set your character for you, and save into disk.

About this time, you'll be called upon to make decisions. First, you choose the attributes of your character: intelligence levels, cunning, and so forth. You can also set such particulars as ego, which may aid your character in later escapes. You'll be called upon to choose your weapons and their size, human nature will make you choose the biggest sword. bow, etc, but this will be no help to your energy levels later -- fighting will be inhibited by using an unwieldy weapon or shield.

Your friendly author set forth into the dank reaches of Apshai as the fearless Rolf Weimann, teutonic freebooter and roustabout. Armed to the hilt with arrows, sword and such, Rolf and I set out to bash some monsters. What we learned in our time together was that adventuring can be fun, and that monsters can bash back sometimes. (I won't spoil the game for you by giving you a blow-by-blow description of the various things you'll see and do, but a few words to the wise will help).

Proceed cautiously through the many rooms of each level. The variety of rooms is staggering. In your travels, you will venture into dank caves, smooth hallways, stagnant monster lairs and catacombes. Computer-generated beasties are counting on

ADVENTURES REVIEW
cont.

your blundering around, and will jump on your frame at the drop of a helmet.

Also, keep an eye on the fatigue and health readings determined by the computer. Ignoring these can be fatal, especially if you stumble across a howling pike-swinging Antman. As in real adventure gaming, you have hit points and such, so know your strengths and guard well your weaknesses.

Keep your treasure hoarding to a minimum, making frequent trips back to the inn to deposit your finds. While you're at it, watch out for traps. Whoever programmed this game must have taken lessons from de sade, because the traps that befall the unwary are grisly: flame trap (roasted alive), ceiling trap (splat), crossbow trap (sproing) and my personal favourite, creeping crud (yaah).

If all of this gives you the idea that I had a ball playing this game, you're right. I've always been a sucker for any game that lets you shoot arrows into people or things, so this was right up my twisted little alley. Rolf and I are good friends now, seasoned in the fire of monster combat.

So it is, too, with the Temple of Apshai Trilogy and me. Fast friends and good ones. So, if you're the armchair-barbarian type, come on in, you have nothing to loose but...

THE ATARI 8-BIT BOOK LENDING LIBRARY

We at TWAUG would like to increase the support we give to our subscribers by starting up an 8-bit book library.

The suggestion for starting the library came from Nir Dary, one of our supporters in the U.S.A. Nir very kindly sent us a large parcel of books to help get us started for which we are extremely grateful. Thank you Nir.

BOOK LENDING LIBRARY
cont.

At present, we have 64 books in the library, and I am busy at the moment cataloguing them all. I was hoping to include a full list in this issue, but we have been kept very busy getting ready for the SAMS show and I was unable to get it ready in time.

It will cost five pound to register with the library, but at the moment, I can't give an amount for the borrowing of books as this will depend on the weight of the book for postage, but charges will be kept as low as possible.

Books can be borrowed for a period of one month, but this can be extended if requested.

If you are interested in joining the library, please get in touch and we will send you a full catalogue and charges. Please write to TWAUG or phone David on 0191/2710086.

BOOKS WANTED.

If anyone has any 8-bit books that they no longer require, but feel that they could be of interest to other users, then we would like to hear from you. Unfortunately we can't offer high prices for books, but we are willing to pay any reasonable price requested. We are also willing to offer to exchange a selection of PD software for books, or maybe you have a book that you would like to donate to the library, if so it would be gratefully received.

FOR SALE SECTION

2 800XLs (standard) complete with power units, £15 each. Postage and packaging for each, £4.50. If you are interested, contact David on 0191/2710086.

QUAD BOARD: The Quad board is an add-on for the Atari 1050 disk drive. Built in to the board are: Laser, Duplicater, Doubler and standard 1050. The board comes

FOR SALE continued

complete with utility disks and full documentation. This is one of the best add-ons that was released for the 1050 disk drive. The Quad board cost £99 mew, I'm selling it for £50 including postage and packing. It has hardly been used, and is in full working order. Contact John on 0191/2626897. Call after 7pm week days, any time weekends.

1050 disk drive fitted with the Happy board. In very good working order, complete with power unit and I/O cable. £50 plus £5 P&P. Contact Mark on 01952/641360, or write to Mark Stinson, 7 Arleston Place, Arleston, Wellington, Telford. TF1 2LT.

MAGAZINES: PAGE SIX issues 1 - 54 (unbroken) = £30.00. PAGE SIX issues 8 -23, 25 - 32, 34, 37, 42 - 43, 49 - 50, 57, 59 - 61, 66 - 67 & 70 = £0,75 each.
ATARI USER Vol.1 No.1 - Vol.4 No.7 (Complete set - unbroken) £25.00.
ATARI USER Vol.1 No.7, Vol.2 No.3, Vol.3 No.8, Vol.2 No.10 - Vol.2 No.12, Vol.3 No.1 - Vol.3 No.12, Vol.4 No.1 - Vol.4 No.2 & Vol.4 No.5 - Vol.4 No.7 = £0.75 each.
ATARI EXPLORER March/April '92 = £0.75.
ATARI CLASSICS Vol.3 No.3 (Sept/Oct '94) £0.75.
MAGAZINE DISKS: PAGE SIX issues 60, 61, 62, 63, 64, 66, 67 & 68 = £1.00 each.
SOFTWARE: Various titles on ROM & Cassettes. Postage on all items will be charged at cost or buyer may collect.
Call Paul on (0161) 737 1946 (Manchester)

SWAPPING SECTION

Nir Darey in the U.S.A, has two 850 interfaces and a MidiMate interface that he would like to swap for anything of interest. If you are interested in any of these items, write to Nir.
His address is: Mr. Nir Dary,
19185 Castlebay LN,
Northridge CA. 91326-1046,
U.S.A.

PEN PALS

A long time 8-bit owner would like to make friends with other Classic owners to swap programming ideas, information, etc. My main interests are utilities, hardware enhancements and DTP. All letters will be answered. Write to Paul, 10 Tenbury Close, Salford, Manchester M6 5BJ.

NOTICE FROM ANG.

We have been informed by ANG in Holland that, due to lack of orders for the BEST of POKEY, there will be no further issues released after issue three.

If you have subscribed to the Best of Pokey, then please get in touch with ANG as soon as possible. John from ANG tells me that he is willing to offer new commercial game titles in place of outstanding issues of the Best of Pokey.

I was very sorry to hear that this disk based magazine would no longer be available as it was one of the best that I've seen. ANG are considering releasing the first three issues of the Best of Pokey in to the public domain, when this does happen, we will of cause let you know.

DISK CONTENT

TWAUG would like to thank "Norman Williamson, Bill Walraven and Mark Watson for submitting the items on the disks. We had so much material for the disks that we decided to give our subscribers a bonus and enclose an extra disk.

These two disks are packed full with Games, Demos and educational programes.

Both disks are run from side A of the disk, you will be presented with a prompt to flip the disk when the program is on side B.

Lets start with disk one: - - BIGABC.BAS, this is an ideal program for the younger members of the

DISK CONTENT continued

family, to teach them the alphabet. When run you are asked to type a letter that is displayed on the screen. It not only displays the character on the screen but creates a much larger character of the one you typed to the screen in upper and lower case.

CIRCLEWW.BAS, this demo prints circles on the screen but follow the prompts to change the display. The second display is also very handy for the younger member of the family, to teach them the time, this display is a working clock and it can be stopped to explain what the time is the hands are pointing to and there are many more displays in that same program.

FISHTXT.BAS, this is also a demo program, it creates a fish with a display of moving text.

FRACTIN.BAS, this is a math program, you are asked if you want addition or subtraction than you type a number and the computer presents you with problem for you to solve.

MAGIC.BAS and USECIRC.BAS, are also demo programs it shows what our 8-bit can do. We have been asked for demo programs and we thought it appropriate to help our members and enclose these programs.

TETRIS3D, is a three dimensional Tetris game. I haven't played it, I only loaded it to screen to see what the display looked and so I am unable to tell you anything more about it.

There are four small FNCI files with text on the disk about Pascal language from Ralph Bradley.

Side B of disk 1 has a number of games and more education programs. Let me start with the education programs first.

PICMATHS, this gives the youngsters the chance to compete against the computer to build a Robot. When you get a sum right a piece of the Robot is added but get the sum wrong and a piece is added to the computers Robot.

GUESSWORDS, when booted, read the instruction you can also get the full selection of words you need for the grid, onto the screen or you can print them out. The aim is to guess the five letter words for the grid, but the computer only gives you the first letter. So it pays to get familiar with the words, unless you are a genius with five letter words.

BILL & BEN is played with a dice, but it's different from a board game. The best thing to do is watch the demo and also read the instructions.

BLOKHEAD, the aim is to retrieve a key on top of a pile of bricks, to get up there you must build a step with other bricks that are falling from the sky. Once you've picked up the key you must move to the next stack for the lock. Again you must build steps with the falling bricks.

FIRE RESCUE, you've got a stretcher and you must catch the people who are jumping out of windows from a high rise building and bounce them onto the waiting ambulance. It's not as easy as it sounds but it is fun.

CASTLE SCOTT, this is adventure game, but I am afraid I am not an adventurer and so have no idea what's it all about.

DISK 2 Side A, has a couple of utilities, the first one being a cassette label printer program. The file name is CASSETTE.BAS.

BAS2BIN.OBJ is a converter program that converts BASIC into BINARY.

CASTAWAY, I believe is a picture adventure.

Side B of this disk is a demo. So have fun with these programs.

DEVICE HANDLERS

by David A. Paterson

From MACAM Disk dated 4/92
[& SLCC #1208]

Everytime you type, save, load, print, format, or do any other operation with your Atari 8-bit computer, you unwittingly use device handlers and the CIO system. This article isn't intended for the beginning BASIC programmer, but rather for the more experienced users looking for information about the inner workings of their system.

The Central Input/Output (CIO) utility was a brilliant innovation by Atari in their first computers. This system, which used a standard method to "talk" with all the peripherals attached to the system; permits you to read or write to just about any device which is hooked up to the system, provided a proper handler for that device is provided. There are five handlers built into your computer:

S the screen. You can write to it; or read from it (with LOCATE in BASIC).

K: the keyboard. You can only read from the keyboard. This device does not display what has been read from the keyboard.

E: the editor. This is a combination of the S: and K: devices, reading information from the keyboard and echoing it to the screen. You can read or write to this device.

P: the printer. This is a write-only device, and sends data to your printer.

C: the cassette, all but forgotten now. You can read or write to it.

Any other handlers, including the Disk handler, have to be loaded into the computer when it powers up. The computer will read in a small piece from D1: (that much code is built in) but the whole disk handler has to be loaded in.

A Handler is made up of three parts. First is an entry in the Handler Address Table (HATABS). This table resides in memory locations 794 to 831, space for up to 12 handlers. Since there are already five built in, that leaves seven open to use. Usually, a handler program begins by looking at HATABS and finding the first blank entry. Each entry is three bytes long; first of all is the identification letter for the device. The next two bytes are the address of the device's handler table.

The second part of the handler is, of course, the device handler table. This table is a series of addresses that tell the system where to look if it has to perform certain actions. There are six routines in the table, in this order:-

OPEN - start I/O with the device.
CLOSE - end I/O with the device.
GET - seek one byte of information from the device. This information will be put into the accumulator.

PUT - send one byte of information to the device. The accumulator holds the byte to be PUT.

STATUS - test the device's current state.

SPECIAL - any other command(s) that this device has.

Each of these entries is two bytes long, and is calculated by taking the address of each routine MINUS one. Why minus one? The system works by

DEVICE HANDLERS continued

tricking the 6502 inside your computer. The two addresses are PusHed from the Accumulator (PHA) onto the stack, then a ReTurn from Subroutine command (RTS) is executed. This command resumes the program at the address at the top of the stack PLUS one. So then, - subtracting one compensates for this.

The last three bytes in the handler are a JMP INIT command, where INIT is the initialization routine for the device.

The final portion of the handler is the commands themselves. These are the commands executed when you OPEN, or CLOSE, or whatever a channel to the device. Before returning from these routines, you must put a value of 1 in the Y-register which indicates that the operation was conducted successfully. In the case of the "SPECIAL" vector, you might have more than one possible special function. The Disk handler, for example, has many "special" functions like LOCK, UNLOCK, ERASE, and FORMAT. The SPECIAL routine has to determine which has been called and go to the appropriate location.

Since the vector tables for all the devices (including the five in ROM) are located in RAM memory, it's possible to change the existing handlers. Atari's XEP-80 device does just that. The XEP handler looks for the S:, E:, and P: handlers, and re-routes their output to the joystick port where the device is attached. Other patches have been written to add extra features to other handlers as well. (more on that later)

This month's (i.e.- the 4/92!) MACAM Journal has (or had!) a bunch of handlers on the program side. They've been taken from Antic and Analog issues. There's a bibliography included at the end of this article if you want more information.

The first handler is the NULL (N:) handler. It's from Antic, and installs a device called N:. This device will take data, and do nothing with it (sort of like a government handler). Running the program NULLHAND.BAS will install the handler. NULLHAND.M65 is the source code.

The next handler is a patch to the P: device. From Antic as well, this handler intercepts the ATASCII character graphics being sent to the printer and replaces them with graphics bytes so that you can actually see them in the listing. PRHAND.BAS will create a file to install this handler patch. The source code is in PRHAND.M65.

The V: device handler comes from ANALOG. This uses your high memory (not extended banks) to store programs or data. The program V.BAS will create a program called V.OBJ which you can load from DOS.

PLEASE NOTE that the three programs just listed CANNOT be run at the same time, as they all use the same memory (page six).

The final handler is the G: device, originally from Analog as well. In short, it's the most useful printer utility that you'll ever see. Written by Charles F. Johnson, of CodeHeads fame, it lists programs, dumps graphics, uses custom character sets, supports multiple sizes of graphics... in only 2.5k of memory. GDRIVER.EXE is the ready-to run version, and GDRIVER.M65 is the source code. It works with Epson compatible printers, and only with printers hooked up to the SIO port (sorry, MIO & Black Box users, and those

DEVICE HANDLERS cont.

with joystick port interfaces). (See SLCC disk #0906 for more info. on G:'s features. - rrs)

BIBLIOGRAPHY

Antic Vol.8 No.2 "Customizing the Atari Operating System Device Handlers" by Bob Martin and Martin Mercorelli.

Analog #31 "V: A Memory Storage Device" by Philip Altman

Analog #35 "G: A Printing Device for Epson / Gemini Printers" by Charles F. Johnson

Atari System Reference Manual by Bob DuHamel

FAMOUS QUOTES THROUGH THE YEARS!

"I found the original of my hell in the world which we inhabit."
- DANTE (1265-1321)

"I live on hope, and that I think do all who come into this world."
- ROBERT BRIDGES (1844-1930)

"The pot calls the kettle black."
- CERVANTES, 'Don Quixote'

"Health is not a condition of matter, but of Mind."
- MARY BAKER EDDY (1821-1910)

"No really great man ever thought himself so."
- WILLIAM HAZLITT (1778-1830)

"To fill the hour-that is happiness."
- RALPH WALDO EMERSON, patron poet of broadcasters.

"Let not your heart be troubled."
- John 14:1

"Men hate more steadily than they love."
- SAMUEL JOHNSON, quoted by Boswell

"How use doth breed a habit in a man."
- SHAKESPEARE 'Two Gentlemen of Verona,' Act V Scene IV

FAMOUS QUOTES cont.

"Genius is mainly an affair of energy"
- MATTHEW ARNOLD (1822-1888)

"Lillies that fester smell far worse than weeds."
- SHAKESPEARE, Sonnet XCIV

"God tempers the cold to the shorn lamb."
- HENRI ESTIENNE (d. 1520)

"There is only one cure for the evils which newly acquired freedom produces and that is more freedom."
- THOMAS MACAULAY (1800-1859)

"Tell me what company thou keepest-and I'll tell thee what thou art."
- CERVANTES, 'Don Quixote'

"When flatterers meet, the Devil goes to dinner."
- JOHN RAY (1627?-1705)

"No one loves the man whom he fears."
- ARISTOTLE (384-322 B.C.E.)

"Fear, like pain, looks and sounds worse than it feels."
- REBECCA WEST (1892-)

"To err is human, to forgive divine."
- ALEXANDER POPE (1688-1744)

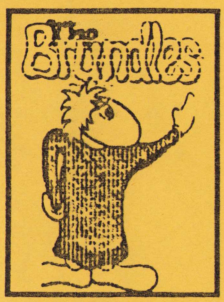
"Experience is the name everyone gives to his mistakes."
- OSCAR WILDE (1854-1900)

"The reward of one duty is the power to fill another."
- GEORGE ELIOT, feminist

"Can one desire too much of a good thing?"
- SHAKESPEARE, 'As You Like It,' Act IV Scene I

"Democracy gives to every man the right to be his own oppressor"
- JAMES RUSSELL LOWELL (1819-1891)

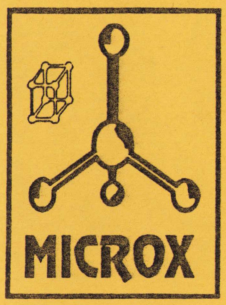
"The fewer desires, the more peace."
- THOMAS WILSON (1663-1755)

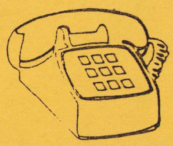


Are You Still Complaining about lack of New software for your Atari 8 bit ?
If you are you don't know about Mail Order From

MICRO-DISCOUNT

265, CHESTER ROAD.
STREETLY.
WEST MIDLANDS.
B74 3EA.
ENGLAND.





TEL 021 353 5730
FAX 021 352 1669

ADAX *HANS KLOSS*

DARKNESS HOUR

TWAUG NEWSLETTER

The Atari Classic Programmer's Club

This is to inform you membership is now only available on yearly subscriptions. The life membership option is no longer available. A member has access to various programmer services, a helpline, regular printed newsletter and discounts off our software?

Current UK membership cost is:

12 months: £6.00

Overseas members most welcome. If you require more details then please send an SAE to the address below. Overseas Atarians, please send two International Reply Coupons (available from your post office) for more information.

Still Available:

Swift Spreadsheet (Standard) - £9.95

Swift Spreadsheet (New) - £12.95

(NOTE: New version includes a revised 40 page A4 manual. For details of the standard version and of the program itself, please refer to the review in issue 67 of New Atari User).

Available Soon:

Menu Print: Still under development and currently undergoing its THIRD rewrite! We apologise for the delay but we expect the programming to be definitely completed by the end of September. More details will be published when ready.

THE ATARI CLASSIC PROGRAMMER'S CLUB
Pen-Tyddyn
Capel Coch
LLangefni
Anglesey, Gwynedd LL77 7UR
Wales

CURRENT NOTES

Helping Atari Owners Through the World of Computing.

Current Notes is published bi-monthly in the U.S.

This magazine has 80 pages full of computer news and very good articles that covers the Atari 8-bit and ST. It comes in full size of 11 inches by 8 1/2 inches.

SUBSCRIPTIONS:

Europe subscriptions is $69 per year.
Bankers drafts made payable to Current Notes

to: CN Subscriptions
122 N. Johnson RD.
Sterling, VA 20164

NOTE: VISA and MasterCard accepted.
Call (703) 450-4761

Editor's Note:
TWAUG is receiving the CN magazine and are always looking forward to it.

PHOENIX.

The new disk based news letter from Ireland, produced by Robert Paden.

PHOENIX a double sided disk, side 'A' will be packed full of text files containing Articles, reviews and much much more. Side 'B' will contain a good selection of PD software.

PHOENIX available from T.W.A.U.G.